

YOUR COMMODORE

AN ARGUS SPECIALIST PUBLICATION

June 1989 £1.50

BASIC



ANALYSER

Basic programs revealed

REVIEWED

► Oxford Basic ► Sketchpad 128

GAMES REVIEWED

► Deadenders ► Middle Earth ► 3D Pool
► Denaris ► The Deep

UNBEATABLE PROGRAMS

► 6510 Assembler ► Line Input
► Help Screen ► Retriever





ONLY \$9.99

CONTENTS

VOLUME 5
NUMBER 9



Super Cycle



Zaxxon



3D Pool

FEATURES

- | | | | |
|--|----|--|----|
| • Ideal Home
Who did it at this famous exhibition | 14 | • Spy Hunter
Popular arcade action | 42 |
| • Oxford Basic
Expand your C64's Basic with this excellent product | 22 | • Demaris
Superb arcade action | 43 |
| • Deadenders
No prizes for guessing what program this is a spoof of! | 32 | • 3D Pool
Computer pool as it should really look | 44 |
| • War in Middle Earth
Adventuring in Tolkien's land | 32 | • Bargain Bucket
The pick of this month's budget games | 45 |
| • Super Cycle
Motor bike racing at its best | 40 | • Disk Editing
Delve further into your disk drive | 50 |
| • The Deep
Under water arcade action | 40 | • Epson SQ-2500
We conclude our look at 24-pin printers | 72 |
| • Grand Prix Circuit
—Try your hand at Formula 1 racing | 41 | • Sketch Pad 128
Does drawing in 80 columns really work? | 74 |
| • Infiltrator
A second chance to fly with Jimbo 'Buky' McGibbons | 42 | | |

REGULARS

- | | | | |
|---|----|---|----|
| • Date Statements
More news from the world of Commodore | 6 | • Extending Basic
Add your own commands to your C64 | 66 |
| • Letters
Your chance to let us know what you think | 10 | • Tech Troubles
Your technical queries answered | 70 |
| • 1st Steps
Help for new programmers | 12 | • Routine Programming
Another handy subroutine to add to your growing library | 76 |
| • Flow of Ideas
Write your own directory reader | 29 | • Software for Sale
If you don't want to type it — Buy it! | 78 |
| • Listings
This month's programs | 55 | | |

PROGRAMS

- | | | | |
|--|----|---|----|
| • Returner
Call up your disk menu with ease | 18 | • Help Screens
Put your C16/+4 HELP key to good use | 38 |
| • Line Input
Improve on Basic's INPUT command | 25 | • ASM Assembler
A powerful assembler for C64 owners | 47 |
| • Program Analysis
Find out just what's happening inside your C64 programs | 34 | | |

Editor: *Steve Cook*
 Deputy Editor:
Paul Whittington
 Technical Editor: *Paul Dan*
 Group Editor:
Mark Webb
 Advertisement Manager:
Paul Kinsman
 Ad-Copy Contact:
Ruth Taylor
 Artist: *Alan Batchelor*
 Designer: *Neil Sweetman*
 Originator: *Alamy*
 Printer: *Chase Webb*

Your Commodore incorporating Year 94 is a monthly magazine appearing on the first Friday of each month. Argus Specialist Publications Limited, Editorial and Advertisement office, Four Commodore, Argus House, Broadway Way, Ipswich, Suffolk IP1 7ST. Telephone: (0447) 666001. Subscription rates upon application to Your Commodore Subscriptions Department, Intense Ltd, 5 River Park Estate, Richmond, Surrey TW9 1HL, U.S.A. Subscription Agent: Wise Owl Worldwide Publications, 4014 West 20th Street, Torrance CA 90505 U.S.A.

**ARGUS
 PRESS
 GROUP**

The contents of this publication including all articles, designs, drawings and programs and all copyright and other intellectual property rights therein belong to Argus Specialist Publications Limited. All rights conferred by the Law of Copyright and other intellectual property rights and by virtue of international copyright conventions are specifically reserved to Argus Specialist Publications Limited and any reproduction requires the prior written consent of the Company. © 1990, Distribution MM Distribution & Logistics Court Road, London SW14 2PG. Printed by Chase Webb, Plymouth. Opinions expressed in reviews are the opinions of the reviewers and not necessarily those of the magazine. While every effort is made to thoroughly check programs published for errors we cannot be held responsible for any errors that do occur.

ISSN 0269-9277

ABC

0269-9277

Free The Spirit

If there's one company above all the others that's trying to make your C128 more like an Amiga, it's Free Spirit Software. This American company seems determined to produce good, high-quality graphics packages for the C128. Their latest offering is Poster Maker 128, with which you can create posters ranging in size up to 15 pages. You need 80 col display and a C128D or C128 with the 64K video Ram upgrade. You can import Basic 8, Starchipad 128 or Spectrum 128 files. There is also an option for reducing, thereby enabling you to perform clip art.

Toolbox: Free Spirit Software, PO Box 121, 38 Noble St, Kestonville, PA 19340 Tel: 215-682-5686.

Spirit of Adventure

Good news for Adventure-Gers. After much public pressure, Incentive Software has rewritten the classic Graphic Adventure Creator. The new release will allow "Stand alone disk accessed" adventures to be produced. This will mean that you can now create adventures of incredible size and complexity, restricted only by disk size. The release date for this exciting product is April 1991. It will retail at £29.95, but existing GAC holders may upgrade for a mere £19.95.

Toolbox: Incentive Software Ltd, Zephyr One, Calver Park, Aldermaston, Berks, RG7 4QH. Tel: 07346 77264.

Data Statements

Chris Payne,
 marketing manager
 at Database, with
 the AMS range



Stop Press!

Database Software have just acquired the rights to two well established DTP packages for the C64 and C128. Both titles originate from AMS. The titles are Stop Press, and Stop Press with mouse, costing £39.99 and £69.99 respectively. Productivity software is a little thin on the ground for the 64/128, so well done Database.

Toolbox: Database Software, Europa House, Aldershot Park, Aldershot, Wokingham, RG30 4NP. Tel: 0625 873888.

Viva Geos

There is no quicker or easier way to present data than with a graph or chart, and GeoChart is a simple and inexpensive way to produce good-looking graphs and charts...etc etc, or so the blurb goes. Bentley Software have added yet another program to their ever growing array of Geos utilities. GeoChart is ideal for anyone that wants professional-looking documents and charts, but can't spare the time needed to plan and draw graphic illustrations.

Like all the Geos appli-

curious, GeoChart is the "What you see, is what you get" format. That is, pull down menus, windows and icons. GeoChart is for the C64 and C128 owners.

Touchline: *Berkley Software, 2128 Shattuck Avenue, Berkeley, California 94704. Tel: 415 644 0881.*



Rob Hubbard - artist in exile

Watch Out

Microprose are about to launch an attack on your wrist with *Navy Seal*, a 64 game packed with action. You are a member of the Elite US Navy force, the sea-air-land commandos, who are experts in sabotage, demolition, reconnaissance and infiltration. So that you don't

Hubba Hubba!

Electronic Arts have remastered that popular game from Bullfinch, *Populous*, by adding a new sound track from Rob Hubbard.

Rob, who left the UK in '88 to work in sunny California, has since been working on a major project. The work involves creating a code package that will allow real soundtracks to be produced. The drivers, we are told, will be available even and over again for any product that requires a soundtrack.

Touchline: *Electronic Arts, Langley Business Centre, 11/49 Station Road, Langley, Berkshire, SL3 8TN. Tel: 0353 69441.*



Make a Stand

Have you noticed how computer furniture never gets a mention? Well, here goes. From MDS of Romel Hampstead comes the MDPS 750 printer stand. Of all the peripherals, printers seem to cause the most positional headaches, but this new stand could solve all your problems (although the £204 price tag may put you off). It is, and I quote "uniquely designed to achieve maximum efficiency, and part of a comprehensive modular desk

system which measures 150mm wide, 710mm high and 750mm deep, is an 18 inch cut-out in the top, making the unit suitable for both top and bottom feed printers". The other good point is the electronic cable management system incorporated into this unit - no more wires and cables need get in the way.

Touchline: *MDS Industries (UK) Ltd, Factory 1, Elm-ers road, Romel Hampstead, HPS 9QS. Tel: 0442 231505.*



ruin the boat, a *Navy Seal* Digital Watch (Right) will be given away with every copy purchased. The release date should be in early April, at an expected cost of £12.95 on disk.

Touchline: *Microprose, 2 Market place, Farnley, Glos, GL8 8DA. Tel: 0683 34126.*

Change At Waterloo

Persons of Restricted Growth everywhere, here's your chance to get your own back! The latest offering from SSI, available via US Gold, allows you to assume the persona of "Sharrs" Bonaparte and change the course of history. Called *Battles of Napoleon* - a construction art, this program allows you to either fight authentic Napoleonic wars, or simply create your own. We are assured that these battles are realistically recreated in every detail, with cavalry charges, bombardments, skirmishes and formations. Priced at £24.99 for the C64, the game offers good value.

Footline: U.S. Gold, Unit 2/3, Halford Way, Halford, Birmingham, B67 7AE. Tel: 021 356 1388.

Special Relationship

News has reached us of a bilateral deal between Kernal and Action Ware.

the outcome of which should prove beneficial to all you game players. The deal means that Kernal will market A/W software here in the U.K., and A/W will market theirs over there. The first product off the press was *Prison*, which was released on March 16th for the Atari ST and Amiga.

The Hammy Hamsters

Fame and fortune can make people do weird and wonderful things. Howard Hughes became a recluse, a certain Mr. Kinn bought a motor factory, and Mr. Wyman managed every man's dream. Now it's the turn of the Darling family. Not content with producing both playable budget software, they have recently broken out into song. To help promote their latest release, *Rock Star*, they've formed a group which, for reasons best known to themselves, is called The Hamsters. Judging by the picture, I know which Hamster I'd like as a pet!

Footline: Code Masters

Software: Co Ltd, Lower Farm House, Stonebridge, Southam, Warwick, CV37 0DL. Tel: 0692 874732.

Amiga Airlines

Buyers of Commodore's new 'Amiga Air Miles Pack' will be receiving free air miles along with their machines. The package offers an Amiga 500, a TV modulator, three new games (*Ripper Rabbit*, *Nebula* and *Star Ray*), comprehensive point packages and 500 free air miles, all for £499.99. The offer will give Commodore customers the chance to book return flights to a number of European cities, including Paris and Amsterdam. All flight vouchers will be valid until March 1991, and can be easily redeemed via the special Commodore Air Miles Hotline, as well as local travel agents.

Footline: Commodore Business Machines (UK) Ltd, Commodore House, The Fairbairn, Gardner Road, Manchester, M6 6LS. Tel: 0623 734616.

Hit The Jackpot At The Commodore Show!

Commodore are giving away £300 worth of computer equipment to the Amiga user with the most innovative and unusual use for his machine. A nationwide search has been launched to discover as many weird and wonderful applications for the Amiga as possible, and entrants will be judged by a panel of experts at the Commodore Computer Show, which opens at the NEC, Birmingham on June 2.

The show itself has been revamped by Database Exhibitions, and will now include seminars on making music, graphics, Commodore magazines (including, of course, PC) and game writing. 'Beat The Author' competitions are also planned, in which punters will be able to take on 'named' authors and try to beat them at their own game. But one of the big highlights of the show will undoubtedly be the innovation competition.

Commodore has already discussed a wide variety of uses for the Amiga, from bird-watching to betting on horses, and has even heard of one user who works out menus for spiders on his machine! If you think you can match these innovative applications, all you need to do is write a brief description - no more than 500 words - and send it in, along with any necessary support material on disk, to:

The Editor,
The Commodore,
Argus House,
Branbury Way,
Hemel Hempstead HP2 8ST.

Entries should reach us before May 31st.



Mailbag

Your chance to air your views in Your Commodore

I have just received a Commodore PET 3602 Computer from a friend. With it was a 8050 Disk Drive and 4022(P) Printer. I was wondering if you or any of your readers had any information on software availability. Please send any information to me, thank you.
R.M. Crook, 19 Grange Road, Bramley Cress, Bolton, BL7 9AU

I was recently given a PET 8032, a 8050 Dual Drive and a 4021 Printer together with four good software packages complete with manuals and dangles, where necessary. I only want to use the PET for wordprocessing, and so I am sending out an appeal, is there anyone out there that has a program for the 8032 that they don't want? If so, please contact me at my address. Thank you.
J. Simpson, 4 Goldenstones Avenue, Dunbar, East Lothian

Come on now everyone, we know there are pretty serious machines, but there must be someone out there who can help!

Dear VC

Is there anything that can be done with old printer

ribbons? I feel that it's a great waste to throw them away.

Could one possibly re-utilize them with ink in some way? If so, which ink should be used, or can I send them somewhere to be done?
J. Gohsen, Luton, Beds

Our reply

The answer, John, is yes. *Printer ribbons can be re-used. If you get a copy of Micro Computer Mart, you will find a firm advertising this service every week.*

Dear VC

With three CDD disks, I have had some loading problems. The disk fits just with your fingers, but rather with the sideways, when they stack magazines on top of each other, I have found that "Validating" the disk beforehand seems to cure this problem.

I don't know what proportion of failures are returned, but this Tip may help in some way.
J.F. Puddick Wood, Tunbridge, Kent

Our reply

Thanks for this helpful note. We do get some returned disks, and many of them are cured by this simple method.

Unfortunately, newspapers and bookshelves do not make these delicate disks any more.

Dear VC

I have been comparing ever since the Commodore 64 was first produced. In this time I have bought various computer magazines, mostly multi-format, and I've noticed various discrepancies between reviews of the same game. For instance, *Tennis* received 50 out of 100 from one, but 97.5 from another.

The trouble with two completely different results is which one do I believe? Do I take the advice of magazine A and stay clear of it, or do I take B's advice and sell my TV so that I can buy it?

The best advice for anyone is to try and get into the reviewer's mind. Some reviewers set their hopes too high, and so if a game doesn't come up to their expectations, they give it a poor write up.

A recent review of *Thunderbirds* is a good example of this. One reviewer of this game said - "The graphics at this point, Level 1 stage 1, were very disappointing - the buildings are just stacked squares".

Personally, I think the graphics are very good. Okay, the first stage on each level is blocky, but you can't really expect hi-res graphics can you? It would be much too slow. Anyway, you've always got stages 2 and 3 in each level to make up for it.

Another annoying item is "Missing". You know the kind of thing - "The Q button doesn't work, so you have to use the P7 key". A great example of this can be found in the same review of *Thunderbirds*. The reviewer said "Unfortunately, it is hard to find any difference between cannon and air-to-

ground rockets." There are two possible reasons for this - (A) His/Her cassette deck/disk drive was faulty or (B) their copy of the game was faulty. These are the only possibilities, as my copy of the game is the opposite.

In conclusion then, what do you do if two reviews differ? The first option is to see if you can have a demonstration of the program, therefore allowing you to make up your own mind. The second option is to simply take a chance.

N. Kingsley, Chichester, West Sussex

Our reply

Thanks for such a long and 'critical' letter. However, I think you're missing the point (as do a lot of people). Reviewers of software do not differ from any other form of reviewer, be it Film, Video, Records, Food, Clothes Etc.

The whole point here is this - the comments passed are the comments of that particular person, NOT the comments of a 'professional critic'. For example, I personally have never liked Elton, but for 1.5 million other users it is the best program ever.

Likewise, for me *Infocom* are the only people that produce decent adventures, but then thousands of people would not agree. Basically, one must remember that a reviewer's comments are his own personal comments. A review only really tells you what the objective is and how the game plays. There is only one person that can say whether it is good or bad, and that is you.

MIXING

Business

WITH

LEISURE

WILL BE
A REAL

EDUCATION

Commodore computer show

Britain's brightest event for Commodore computer users is back! And there's more to see than ever before.

This show has five main themes covering some of the major uses to which Commodore machines are put. There are over 70 key companies who will be exhibiting their latest products, which means that just about everything that's new in the Commodore world will be on show!

Business

Many companies will be demonstrating their latest software and hardware specially designed to release the full business potential of Commodore computers.

As well as products for the C64 and Amiga series, you'll be able to try out applications for the price-leading Commodore PC-compatible micros.

And you'll also be able to attend seminars covering all aspects of using Commodore micros in your business.

Leisure

The C64 and Amiga computers are the most powerful 8 and 16 bit micros for producing fast-action arcade quality games. The range of new software on show

Novotel Exhibition Complex,
Hammersmith, London W6
Friday to Sunday
June 2 to 4

Visiting Friday & Saturday: 10am-6pm Sunday

will demonstrate how these machines' power is continually being stretched, producing faster and even more addictive games with superb graphics.

If you're a keen game player, you'll find there's so much on offer at the show you're guaranteed a real treat!

Education

Commodore micros are now used as educational tools all over the country. With the development of BBC Basic on the Amiga and the advent of Desktop Video (combining TV pictures with text and graphics), the range of educational applications is endless.

At the show you'll see how the latest software

packages are making real breakthroughs in the educational sector, and be able to try them out for yourself.

Special Events

As well as special events and presentations, you'll also be able to meet some of your favourite software stars, and maybe get a chance to talk with them about how they use micros in their work.

So for a great day out, whether you want to see what the future holds for Commodore computers, to buy the latest software or to get advice on specific applications, the Commodore show is the place to go. And if you want to be there today, we'll knock £1 off the price of each ticket!

For the first time we are offering a family ticket for just £11 allowing entry for two adults and two children - saving up to £7 off the usual entry price!

How To Get There

By Underground: Hammersmith (Piccadilly, Metropolitan & District)

By Bus: 260, 744, 710, 260, 30, 52, 73, 74

Car parking facilities available at the Novotel.

Advanced ticket order

Commodore
computer show

PO BOX 110, Commodore Show Sales,
PO Box 2, Chessington Park,
Surrey (Windsor) LE6 9BA.

Please supply:

☐ Adult tickets at £4 (save £1) £

☐ Under-16s tickets at £3 (save £1) £

☐ Family ticket at £11 (save £7) £

Total £

☐ Cheque payable to Commodore Exhibitions.

☐ Please debit my Access/Visa card no.

Entry date:

Signature

Authorisation at show

£1 (refund)

£2.50 (refund 10%)

Advanced ticket orders

must be received by

Wednesday, May 24

Name: _____

Address: _____

Postcode: _____

Phone: 01844 511111, 01844 511111, 01844 511111

Only UK residents can use this order form.

For further information, please contact Commodore Show Sales, PO Box 110, Chessington Park, Surrey (Windsor) LE6 9BA.

Please quote credit card number and full address.



How to make sense of Listings By Norman Doyle

Typing listings can be much more of a trauma than it need be. At *Your Commodore*, we are aware that sometimes the process seems difficult or even nonsensical. Our Listings with their Syntax Checker and special codings may ease the way, but sometimes more support is required.

Many programs are listed in several parts and each part uses the same line numbers. How is this possible and how do you set about typing them in?

First of all it should be explained that a set of these programs form a suite which operate in unison. The way to use them is to type in the first program and save it. Now the computer should be cleared by entering NEW or even by switching the computer off and on.

Proceeding to the second listing, the program is again typed in and saved. If you're storing the programs on tape, you should ensure that this program is saved immediately after the first one. For disk users, it doesn't matter where it's saved as long as it's

on the same disk as the first program. After checking that all has been saved safely, the machine is again refreshed using NEW or the power switch.

Any subsequent programs should be loaded in the same way as the second listing and the result will be a string of programs. One word of caution which should be heeded is that the programs should be saved under the name given in *Your Commodore* because each program occasionally uses one of these names to load in the next program in the string.

Suite Confusion

One question which we've often asked is why these multipart programs are needed. The answer is that they are a memory-saving or time-saving measure.

Many programs use redefined characters, machine code patches and sprites. The information for this has to be poked into position because they rarely occupy memory locations which Basic uses. This difficulty can be overcome in either of two ways.

The data can be poked into position each time the program is loaded and forms an integral part of the program. This has the dual disadvantage of the time the program takes to move the data in but, more importantly, reduces the amount of space for the master program itself.

Another way of achieving the same result is to use a series of set-up programs before the main program is loaded. The set-ups are loaded and run in turn and each one pokes information into memory where it cannot be touched by any subsequent loading actions. Consequently, once the program has done its job it is no longer needed and the program can even be removed by typing in NEW without ruining the effect of the main program when it is eventually loaded.

One thing that you cannot do is to switch the computer off and on again! This action totally clears the memory, including the data which has been poked in.

When the first program has done its job, the next program is loaded normally and run. Now there are two

blocks of data locked away safely. This continues until the final program is loaded. This is the main program which holds the key to unloading and using the data which the previous programs positioned. The fact that each of the programs may have contained the same line numbers is totally irrelevant because it's what each program does and not how it does it. It's a little like receiving a birthday card and an electricity bill - they both carry the same address but the effect they have on the mind is totally different!

Gift Boxes

Although this system solves the problem of memory space, it does not save time and a long program can seem to take forever to push the data into position. One way to overcome this is to create a program which not only pushes the data into position, but also saves the pure data block afterwards.

These 'downloader' programs, or Basic loaders to give them their correct name, are quite common and need treating separately. The process can be

quite complicated so we'll take the case of a single program first.

As usual, the program is typed in and saved before running it. Next a new tape or disk should be placed in the storage device. Now when the program runs it will save the data back onto the new tape but in a different form. The new program will be loaded directly into memory without the need for a Basic program to push it there. These programs usually have to be started with a SYS command and, once checked, the program that was originally typed in can be forgotten and erased.

The Basic program is rather like a gift box which becomes useless once the contents have been removed.

A program of this kind which is separated into parts is immediately recognisable because the master program is the first in the chain. If this is in Basic it should be typed in and saved. If it is a Basic loader it must be treated differently.

Spot The Loader

A Basic loader can be recognised by

the terms of DATA statements which must be typed in. Another indicator is the character of the actual program, including the DATA statements, which often only contain a FOR...NEXT loop separated by complex POKE commands.

If the main program proves to be a Basic loader it should be typed in and saved but when it is run a separate disk or tape should be already in the same device.

After dealing with the main program, any subsequent parts should be typed in and this routine saved to the first tape or disk. On running, the resultant program should be saved on the second tape immediately after the preceding part.

Basically, the rule to remember is that any programs which have duplicated line numbers cannot reside in memory at the same time. They should be typed in and saved separately according to the instructions given in the relevant Your Commodore article or the REM statements in the listings themselves. Stick to this rule and you'll never go wrong...well, rarely!

TELETEXT

A world of information at your fingertips



Teletext on your 64 or 128 brings you the very latest information fast! Available a Teletext TV you can print pages like today's TV, turn a recipe to film or cassette. The ability to access Teletext data from your own programs provides endless possibilities. Contact us! Oracle provides hundreds of pages of news, share prices, weather and mail reports, even bargain holidays plus much much more.

The Microtext Teletext Adapter fits neatly on the rear port, just connect it to the Tuner and plug in an aerial or the Adapter alone may be connected to the VIDEO OUT socket of a video recorder.

The Microtext Adapter is only £29.95. Adapter and Tuner just £29.95 including VAT and p.p.h.

NEW Upgrader

The Upgrader allows your 486 Microtext Adapter to be connected to the Amiga and comes complete with Amiga software for only £24.95.

MICROTEXT
Dept. PC 7, 7 Beldy Close, Horden, Gateshead, North Coes DA6
Telephone: (0775) 587894

OFFICIAL COMMODORE/AMIGA DEALER

COMMODORE	
Amiga 500 - 1 Year Limited Warranty	£299.00
Amiga 500 - 3 Year Limited Warranty	£349.00
Amiga 500 - 5 Year Limited Warranty	£399.00
Amiga 500 - 7 Year Limited Warranty	£449.00
Amiga 500 - 9 Year Limited Warranty	£499.00
Amiga 500 - 11 Year Limited Warranty	£549.00
Amiga 500 - 13 Year Limited Warranty	£599.00
Amiga 500 - 15 Year Limited Warranty	£649.00
Amiga 500 - 17 Year Limited Warranty	£699.00
Amiga 500 - 19 Year Limited Warranty	£749.00
Amiga 500 - 21 Year Limited Warranty	£799.00
Amiga 500 - 23 Year Limited Warranty	£849.00
Amiga 500 - 25 Year Limited Warranty	£899.00
Amiga 500 - 27 Year Limited Warranty	£949.00
Amiga 500 - 29 Year Limited Warranty	£999.00
Amiga 500 - 31 Year Limited Warranty	£1049.00
Amiga 500 - 33 Year Limited Warranty	£1099.00
Amiga 500 - 35 Year Limited Warranty	£1149.00
Amiga 500 - 37 Year Limited Warranty	£1199.00
Amiga 500 - 39 Year Limited Warranty	£1249.00
Amiga 500 - 41 Year Limited Warranty	£1299.00
Amiga 500 - 43 Year Limited Warranty	£1349.00
Amiga 500 - 45 Year Limited Warranty	£1399.00
Amiga 500 - 47 Year Limited Warranty	£1449.00
Amiga 500 - 49 Year Limited Warranty	£1499.00
Amiga 500 - 51 Year Limited Warranty	£1549.00
Amiga 500 - 53 Year Limited Warranty	£1599.00
Amiga 500 - 55 Year Limited Warranty	£1649.00
Amiga 500 - 57 Year Limited Warranty	£1699.00
Amiga 500 - 59 Year Limited Warranty	£1749.00
Amiga 500 - 61 Year Limited Warranty	£1799.00
Amiga 500 - 63 Year Limited Warranty	£1849.00
Amiga 500 - 65 Year Limited Warranty	£1899.00
Amiga 500 - 67 Year Limited Warranty	£1949.00
Amiga 500 - 69 Year Limited Warranty	£1999.00
Amiga 500 - 71 Year Limited Warranty	£2049.00
Amiga 500 - 73 Year Limited Warranty	£2099.00
Amiga 500 - 75 Year Limited Warranty	£2149.00
Amiga 500 - 77 Year Limited Warranty	£2199.00
Amiga 500 - 79 Year Limited Warranty	£2249.00
Amiga 500 - 81 Year Limited Warranty	£2299.00
Amiga 500 - 83 Year Limited Warranty	£2349.00
Amiga 500 - 85 Year Limited Warranty	£2399.00
Amiga 500 - 87 Year Limited Warranty	£2449.00
Amiga 500 - 89 Year Limited Warranty	£2499.00
Amiga 500 - 91 Year Limited Warranty	£2549.00
Amiga 500 - 93 Year Limited Warranty	£2599.00
Amiga 500 - 95 Year Limited Warranty	£2649.00
Amiga 500 - 97 Year Limited Warranty	£2699.00
Amiga 500 - 99 Year Limited Warranty	£2749.00
Amiga 500 - 101 Year Limited Warranty	£2799.00
Amiga 500 - 103 Year Limited Warranty	£2849.00
Amiga 500 - 105 Year Limited Warranty	£2899.00
Amiga 500 - 107 Year Limited Warranty	£2949.00
Amiga 500 - 109 Year Limited Warranty	£2999.00
Amiga 500 - 111 Year Limited Warranty	£3049.00
Amiga 500 - 113 Year Limited Warranty	£3099.00
Amiga 500 - 115 Year Limited Warranty	£3149.00
Amiga 500 - 117 Year Limited Warranty	£3199.00
Amiga 500 - 119 Year Limited Warranty	£3249.00
Amiga 500 - 121 Year Limited Warranty	£3299.00
Amiga 500 - 123 Year Limited Warranty	£3349.00
Amiga 500 - 125 Year Limited Warranty	£3399.00
Amiga 500 - 127 Year Limited Warranty	£3449.00
Amiga 500 - 129 Year Limited Warranty	£3499.00
Amiga 500 - 131 Year Limited Warranty	£3549.00
Amiga 500 - 133 Year Limited Warranty	£3599.00
Amiga 500 - 135 Year Limited Warranty	£3649.00
Amiga 500 - 137 Year Limited Warranty	£3699.00
Amiga 500 - 139 Year Limited Warranty	£3749.00
Amiga 500 - 141 Year Limited Warranty	£3799.00
Amiga 500 - 143 Year Limited Warranty	£3849.00
Amiga 500 - 145 Year Limited Warranty	£3899.00
Amiga 500 - 147 Year Limited Warranty	£3949.00
Amiga 500 - 149 Year Limited Warranty	£3999.00
Amiga 500 - 151 Year Limited Warranty	£4049.00
Amiga 500 - 153 Year Limited Warranty	£4099.00
Amiga 500 - 155 Year Limited Warranty	£4149.00
Amiga 500 - 157 Year Limited Warranty	£4199.00
Amiga 500 - 159 Year Limited Warranty	£4249.00
Amiga 500 - 161 Year Limited Warranty	£4299.00
Amiga 500 - 163 Year Limited Warranty	£4349.00
Amiga 500 - 165 Year Limited Warranty	£4399.00
Amiga 500 - 167 Year Limited Warranty	£4449.00
Amiga 500 - 169 Year Limited Warranty	£4499.00
Amiga 500 - 171 Year Limited Warranty	£4549.00
Amiga 500 - 173 Year Limited Warranty	£4599.00
Amiga 500 - 175 Year Limited Warranty	£4649.00
Amiga 500 - 177 Year Limited Warranty	£4699.00
Amiga 500 - 179 Year Limited Warranty	£4749.00
Amiga 500 - 181 Year Limited Warranty	£4799.00
Amiga 500 - 183 Year Limited Warranty	£4849.00
Amiga 500 - 185 Year Limited Warranty	£4899.00
Amiga 500 - 187 Year Limited Warranty	£4949.00
Amiga 500 - 189 Year Limited Warranty	£4999.00
Amiga 500 - 191 Year Limited Warranty	£5049.00
Amiga 500 - 193 Year Limited Warranty	£5099.00
Amiga 500 - 195 Year Limited Warranty	£5149.00
Amiga 500 - 197 Year Limited Warranty	£5199.00
Amiga 500 - 199 Year Limited Warranty	£5249.00
Amiga 500 - 201 Year Limited Warranty	£5299.00
Amiga 500 - 203 Year Limited Warranty	£5349.00
Amiga 500 - 205 Year Limited Warranty	£5399.00
Amiga 500 - 207 Year Limited Warranty	£5449.00
Amiga 500 - 209 Year Limited Warranty	£5499.00
Amiga 500 - 211 Year Limited Warranty	£5549.00
Amiga 500 - 213 Year Limited Warranty	£5599.00
Amiga 500 - 215 Year Limited Warranty	£5649.00
Amiga 500 - 217 Year Limited Warranty	£5699.00
Amiga 500 - 219 Year Limited Warranty	£5749.00
Amiga 500 - 221 Year Limited Warranty	£5799.00
Amiga 500 - 223 Year Limited Warranty	£5849.00
Amiga 500 - 225 Year Limited Warranty	£5899.00
Amiga 500 - 227 Year Limited Warranty	£5949.00
Amiga 500 - 229 Year Limited Warranty	£5999.00
Amiga 500 - 231 Year Limited Warranty	£6049.00
Amiga 500 - 233 Year Limited Warranty	£6099.00
Amiga 500 - 235 Year Limited Warranty	£6149.00
Amiga 500 - 237 Year Limited Warranty	£6199.00
Amiga 500 - 239 Year Limited Warranty	£6249.00
Amiga 500 - 241 Year Limited Warranty	£6299.00
Amiga 500 - 243 Year Limited Warranty	£6349.00
Amiga 500 - 245 Year Limited Warranty	£6399.00
Amiga 500 - 247 Year Limited Warranty	£6449.00
Amiga 500 - 249 Year Limited Warranty	£6499.00
Amiga 500 - 251 Year Limited Warranty	£6549.00
Amiga 500 - 253 Year Limited Warranty	£6599.00
Amiga 500 - 255 Year Limited Warranty	£6649.00
Amiga 500 - 257 Year Limited Warranty	£6699.00
Amiga 500 - 259 Year Limited Warranty	£6749.00
Amiga 500 - 261 Year Limited Warranty	£6799.00
Amiga 500 - 263 Year Limited Warranty	£6849.00
Amiga 500 - 265 Year Limited Warranty	£6899.00
Amiga 500 - 267 Year Limited Warranty	£6949.00
Amiga 500 - 269 Year Limited Warranty	£6999.00
Amiga 500 - 271 Year Limited Warranty	£7049.00
Amiga 500 - 273 Year Limited Warranty	£7099.00
Amiga 500 - 275 Year Limited Warranty	£7149.00
Amiga 500 - 277 Year Limited Warranty	£7199.00
Amiga 500 - 279 Year Limited Warranty	£7249.00
Amiga 500 - 281 Year Limited Warranty	£7299.00
Amiga 500 - 283 Year Limited Warranty	£7349.00
Amiga 500 - 285 Year Limited Warranty	£7399.00
Amiga 500 - 287 Year Limited Warranty	£7449.00
Amiga 500 - 289 Year Limited Warranty	£7499.00
Amiga 500 - 291 Year Limited Warranty	£7549.00
Amiga 500 - 293 Year Limited Warranty	£7599.00
Amiga 500 - 295 Year Limited Warranty	£7649.00
Amiga 500 - 297 Year Limited Warranty	£7699.00
Amiga 500 - 299 Year Limited Warranty	£7749.00
Amiga 500 - 301 Year Limited Warranty	£7799.00
Amiga 500 - 303 Year Limited Warranty	£7849.00
Amiga 500 - 305 Year Limited Warranty	£7899.00
Amiga 500 - 307 Year Limited Warranty	£7949.00
Amiga 500 - 309 Year Limited Warranty	£7999.00
Amiga 500 - 311 Year Limited Warranty	£8049.00
Amiga 500 - 313 Year Limited Warranty	£8099.00
Amiga 500 - 315 Year Limited Warranty	£8149.00
Amiga 500 - 317 Year Limited Warranty	£8199.00
Amiga 500 - 319 Year Limited Warranty	£8249.00
Amiga 500 - 321 Year Limited Warranty	£8299.00
Amiga 500 - 323 Year Limited Warranty	£8349.00
Amiga 500 - 325 Year Limited Warranty	£8399.00
Amiga 500 - 327 Year Limited Warranty	£8449.00
Amiga 500 - 329 Year Limited Warranty	£8499.00
Amiga 500 - 331 Year Limited Warranty	£8549.00
Amiga 500 - 333 Year Limited Warranty	£8599.00
Amiga 500 - 335 Year Limited Warranty	£8649.00
Amiga 500 - 337 Year Limited Warranty	£8699.00
Amiga 500 - 339 Year Limited Warranty	£8749.00
Amiga 500 - 341 Year Limited Warranty	£8799.00
Amiga 500 - 343 Year Limited Warranty	£8849.00
Amiga 500 - 345 Year Limited Warranty	£8899.00
Amiga 500 - 347 Year Limited Warranty	£8949.00
Amiga 500 - 349 Year Limited Warranty	£8999.00
Amiga 500 - 351 Year Limited Warranty	£9049.00
Amiga 500 - 353 Year Limited Warranty	£9099.00
Amiga 500 - 355 Year Limited Warranty	£9149.00
Amiga 500 - 357 Year Limited Warranty	£9199.00
Amiga 500 - 359 Year Limited Warranty	£9249.00
Amiga 500 - 361 Year Limited Warranty	£9299.00
Amiga 500 - 363 Year Limited Warranty	£9349.00
Amiga 500 - 365 Year Limited Warranty	£9399.00
Amiga 500 - 367 Year Limited Warranty	£9449.00
Amiga 500 - 369 Year Limited Warranty	£9499.00
Amiga 500 - 371 Year Limited Warranty	£9549.00
Amiga 500 - 373 Year Limited Warranty	£9599.00
Amiga 500 - 375 Year Limited Warranty	£9649.00
Amiga 500 - 377 Year Limited Warranty	£9699.00
Amiga 500 - 379 Year Limited Warranty	£9749.00
Amiga 500 - 381 Year Limited Warranty	£9799.00
Amiga 500 - 383 Year Limited Warranty	£9849.00
Amiga 500 - 385 Year Limited Warranty	£9899.00
Amiga 500 - 387 Year Limited Warranty	£9949.00
Amiga 500 - 389 Year Limited Warranty	£9999.00
Amiga 500 - 391 Year Limited Warranty	£10049.00
Amiga 500 - 393 Year Limited Warranty	£10099.00
Amiga 500 - 395 Year Limited Warranty	£10149.00
Amiga 500 - 397 Year Limited Warranty	£10199.00
Amiga 500 - 399 Year Limited Warranty	£10249.00
Amiga 500 - 401 Year Limited Warranty	£10299.00
Amiga 500 - 403 Year Limited Warranty	£10349.00
Amiga 500 - 405 Year Limited Warranty	£10399.00
Amiga 500 - 407 Year Limited Warranty	£10449.00
Amiga 500 - 409 Year Limited Warranty	£10499.00
Amiga 500 - 411 Year Limited Warranty	£10549.00
Amiga 500 - 413 Year Limited Warranty	£10599.00
Amiga 500 - 415 Year Limited Warranty	£10649.00
Amiga 500 - 417 Year Limited Warranty	£10699.00
Amiga 500 - 419 Year Limited Warranty	£10749.00
Amiga 500 - 421 Year Limited Warranty	£10799.00
Amiga 500 - 423 Year Limited Warranty	£10849.00
Amiga 500 - 425 Year Limited Warranty	£10899.00
Amiga 500 - 427 Year Limited Warranty	£10949.00
Amiga 500 - 429 Year Limited Warranty	£10999.00
Amiga 500 - 431 Year Limited Warranty	£11049.00
Amiga 500 - 433 Year Limited Warranty	£11099.00
Amiga 500 - 435 Year Limited Warranty	£11149.00
Amiga 500 - 437 Year Limited Warranty	£11199.00
Amiga 500 - 439 Year Limited Warranty	£11249.00
Amiga 500 - 441 Year Limited Warranty	£11299.00
Amiga 500 - 443 Year Limited Warranty	£11349.00
Amiga 500 - 445 Year Limited Warranty	£11399.00
Amiga 500 - 447 Year Limited Warranty	£11449.00
Amiga 500 - 449 Year Limited Warranty	£11499.00
Amiga 500 - 451 Year Limited Warranty	£11549.00
Amiga 500 - 453 Year Limited Warranty	£11599.00
Amiga 500 - 455 Year Limited Warranty	£11649.00
Amiga 500 - 457 Year Limited Warranty	£11699.00
Amiga 500 - 459 Year Limited Warranty	£11749.00
Amiga 500 - 461 Year Limited Warranty	£11799.00
Amiga 500 - 463 Year Limited Warranty	£11849.00
Amiga 500 - 465 Year Limited Warranty	£11899.00
Amiga 500 - 467 Year Limited Warranty	£11949.00
Amiga 500 - 469 Year Limited Warranty	£11999.00
Amiga 500 - 471 Year Limited Warranty	£12049.00
Amiga 500 - 473 Year Limited Warranty	£12099.00
Amiga 500 - 475 Year Limited Warranty	£12149.00
Amiga 500 - 477 Year Limited Warranty	£12199.00
Amiga 500 - 479 Year Limited Warranty	£12249.00
Amiga 500 - 481 Year Limited Warranty	£12299.00
Amiga 500 - 483 Year Limited Warranty	£12349.00
Amiga 500 - 485 Year Limited Warranty	£12399.00
Amiga 500 - 487 Year Limited Warranty	£12449.00
Amiga 500 - 489 Year Limited Warranty	£12499.00
Amiga 500 - 491 Year Limited Warranty	£12549.00
Amiga 500 - 493 Year Limited Warranty	£12599.00
Amiga 500 - 495 Year Limited Warranty	£12649.00
Amiga 500 - 497 Year Limited Warranty	£12699.00
Amiga 500 - 499 Year Limited Warranty	£12749.00
Amiga 500 - 501 Year Limited Warranty	£12799.00
Amiga 500 - 503 Year Limited Warranty	£12849.00
Amiga 500 - 505 Year Limited Warranty	£12899.00
Amiga 500 - 507 Year Limited Warranty	£12949.00
Amiga 500 - 509 Year Limited Warranty	£12999.00
Amiga 500 - 511 Year Limited Warranty	£13049.00
Amiga 500 - 513 Year Limited Warranty	£13099.00
Amiga 500 - 515 Year Limited Warranty	£13149.00
Amiga 500 - 517 Year Limited Warranty	£13199.00
Amiga 500 - 519 Year Limited Warranty	£13249.00
Amiga 500 - 521 Year Limited Warranty	£13299.00
Amiga 500 - 523 Year Limited Warranty	£13349.00
Amiga 500 - 525 Year Limited Warranty	£13399.00
Amiga 500 - 527 Year Limited Warranty	£13449.00
Amiga 500 - 529 Year Limited Warranty	£13499.00
Amiga 500 - 531 Year Limited Warranty	£13549.00
Amiga 500 - 533 Year Limited Warranty	£13599.00
Amiga 500 - 535 Year Limited Warranty	£13649.00
Amiga 500 - 537 Year Limited Warranty	£13699.00
Amiga 500 - 539 Year Limited Warranty	£13749.00
Amiga 500 - 541 Year Limited Warranty	£13799.00
Amiga 500 - 543 Year Limited Warranty	£13849.00
Amiga 500 - 545 Year Limited Warranty	£13899.00
Amiga 500 - 547 Year Limited Warranty	£13949.00
Amiga 500 - 549 Year Limited Warranty	£13999.00
Amiga 500 - 551 Year Limited Warranty	£14049.00
Amiga 500 - 553 Year Limited Warranty	£14099.00
Amiga 500 -	

The Case of the Electronic Home

'Holmes, I've got a letter here from a distraught magazine editor asking us to investigate the future of the electronic home, but I don't know how to go about the job.' 'Well Watson, you can always pop down to the furniture exhibition at Earl's Court and see what's new.' 'That's Ideal Holmes! (Oh God - Ed)

So it was that our two intrepid detectives went off in search of clues as to how long it would be before everything in the house could be operated at the flick of a switch.

When they arrived at the show though, they were in for something of a surprise. Whereas even a couple of years before, every machine was being heavily promoted as the all singing, all dancing model with more lights and buttons than you know what to do with, this year's displays were much more minimalist in design. It was as if it was assumed that the machine could do everything that was asked of it, and it was simply left at that.

'There are two distinct problems here' said Holmes. 'To start with, you will notice that just about every machine, from cameras to washing machines, has chips in it, which in turn gives the end user access to a plethora of functions. But although the customer may appreciate having all these extra facilities, he doesn't actually use many of them. After all, just about every single compact disc player lets you program the tracks in any order that you want. But who in their right mind is going to mess about with the order of the movements in a Mozart symphony?'

'The second problem is that nothing is compatible unless you stick to just one manufacturer. So the customer ends up with separate remote control handsets for his television, video and hi-fi. None of them will turn on his dishwasher, and he can only use them over a limited range. What is needed is some all-powerful control system.

Leaving aside his relentless search for the dreaded Moriarty, Mr Sherlock Holmes and his old friend Dr Watson visit the Ideal Homes exhibition and discover that 221 B Baker Street is not all that it should be...

By Gordon Hamlett



'I notice that all the display houses have a study' commented Watson. 'The idea of people working more and more from home perishes, even though there is little evidence to suggest that it is actually happening. It seems that computers are still regarded as toys unless they possess those three magic letters - IBM. Moderns still have not caught on as they have done in

America. Instead, no home should be without its own personal fax machine.'

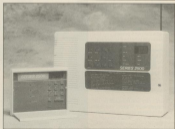
'If we are not very careful Watson, you and I could soon be out of a job. The one major area of expansion in the electronic field is in security systems. I see that Modern Alarms are offering a wireless programmable system - easy to install, no messing up the decoration, and it can be taken



with you when you move. And every type of alarm imaginable is here — magnetic contacts, heat, radar, sound, vibration, breaking glass, infra red beams and close circuit wiring. There are even personal alarm buttons.

'How is the enterprising burglar supposed to get past that little lot, and if there are no criminals, then what are we to do? Ah, here is something that would fool even me. A machine that emits artificial barking noises. Remember that case of the dog that didn't bark in the night? This device would have totally messed up that story.'

'I can find no trace of robotics here Holmes. It looks as if we will have to put up with Mrs Hudson's housekeeping for at least another year. This is one field that has definitely not progressed as quickly as everybody anticipated, although I will have reasons that two of the biggest fast food chains are battling to be the first



to get their customers served by automations.

"At last, Watson, I think that we have found what we were looking for. A system that will control every domestic appliance in the house and which can be operated either from the central processing unit, from a handset or, more importantly, via a telephone link. Just think of it. Being able to phone home if we are out on a case and instruct the video to record *Crimewatch* and *Police 1*."

"I see that the system - *Credna* from *Creda* - works by means of a series of transmitters placed in every appliance, it then communicates with the central processing unit using existing mains-circuitry. This c.p.u. has a large display panel as well, so that you keep addresses and diary dates as well as being able to check up on share prices and train times. That would have been useful in the *Baskerville* case."

"We would be able to control the temperature in every room in the house, turning it on and when we are fit. Lighting could be similarly controlled, ranging from individual lamps upwards. Imagine being able to turn out the kitchen light from your bed when you had forgotten to do so. All your aforementioned security devices could be checked easily, although it would lessen the impact of my detective powers if I knew beforehand exactly who had turned up at my front door."

"Turning on the cooker, doing the washing in the middle of the night and arranging for records of my beloved violin music to start playing as soon as I walked through the door. And all available in the next couple of years for about a thousand pounds."

"This has got to be the way forward, eh Watson? This is where the future of the electronic house lies. It doesn't matter what make of appliances you've got - they can all be controlled from the one system. I would guess that we will be hearing of many more such control systems in the next few years, each offering more and more features and with the price coming down all the time."

"There's only one thing left for us to investigate now Watson, and that is to find out what is on the other side of that yellow door." What's so special about the yellow door Holmes? "It's a lemon entry my dear Watson! (After some, *Hammer*, no more please - a distraught Ed).



Quality. Designed from the inside out.

The Business Series



Value

24

Letter



Simplicity



Folds



Paper Folding



Speed



Compatibility



Support

For state-of-the-art printing, the multi-ton LC24-85 gives you value for money as well as the superior quality of a 24-pin printhead.

The LC24-85 is compatible with almost every computer's needs, whatever the system. It can emulate most industry standards with ease - a standard feature that we think should always be included. For convenience, there is even the option of slow and serial emulations.

Printing the LC24-85 to work couldn't be simpler. At the mere touch of a button on the front control panel you can choose any one of eight variable letter quality fonts, determine your



pitch, or simply put the paper part into action. (The choice of convenient DIP switches is simpler

you). As with all our printers, it offers both double and quad-high-pin for striking leaders.

Powerful 170 characters in one second and that's just the rate dial. The LC24-85 will also perform word perfectly at rapid 51 cps producing exceptional letter quality for printed correspondence and important documents. And while you print, the 16,000 buffer feeds you and your PC to get on with other things.

No other 24-pin printer in this price range can match it. Features like its generous toner buffer, expandable to 196, built-in paper feeder and revolutionary continuous paper feeding facility are all included as standard. Another impressive feature you won't find in an optional extra is the LC24-85's versatile range of print variations. All fonts come with the option of shadow, outline and even shadowed outline to form alternative styled text.

Behind every Star printer there's a guarantee of superb reliability together with a comprehensive back-up service for peace of mind. Should you need any help at any time, you can rely on our national three-level support system which operates throughout the dealer network and is backed by the finest distributors in the country.



THE
star
COMPUTER PRINTERS

Star Machines Ltd., Ltd.,
Crown House, 40 Colindale Avenue, London NW9 1BS. Telephone: 01-540 1600.

A Division of Star Machines Co., Ltd., Japan.

Please send me all the information I'll need. (Alternatively, just call. Ref: 01-540 1679).

Name _____ Company _____

Address _____

Postcode _____ Telephone _____

LC24-85

All trade marks acknowledged.



Public sector customers may purchase at preferential terms from PPSW3. Contact 0603 6891250.

Returmer

Return to the menu at any time with this ingenious program

The problem of keeping records of what's on your disk was addressed in *Commodore Disk User* Vol. 1, No. 3 (Jan/Feb 88) by Menu Maker. This useful utility allowed you to load any program direct from the Menu, but to return to MENU you had to add the load statement to each program on the file, which is easy enough to do, but usually means asking check questions in and having to end the running up of the program. You could also do it by pressing Run Stop and retyping "Menu", but that's rather a waste of time.

The program Returmer allows you to leave a program at any point or time, and return to the menu to select an alternative program without all the problems outlined above.

On loading and activating the 'RETURMER' program code, the first job that it does is to set up the 'RESTORE' key, and automatically load a MENU.

Obviously, for the program to work correctly, you must have a 'MENU' program on your disk. We have published quite a few such programs in the past. Alternatively, you could write one of your own. To get the best results, it is advisable to have at least two, if not more, programs on your menu.

Once you have your menu program running, you select which program you want to normal. On running the selected program, you can press the 'RESTORE' key at any time to return to your MENU program.

If the returmer won't work

There are two reasons why Returmer may not work:

1. The program in memory is using the NMI interrupt, or it is restoring the interrupt vectors to normal values. You need to alter your program (if possible) so that it's not using the NMI interrupt.

N.B. The NMI interrupt values are located at 790-793 (\$318-\$31B).

2. The memory locations where 'Returmer' is located is being used by the program being run. See solution below.

Program checker

This program will provide you with alternative memory locations to hold the 'Returmer' program. Load 'CHECKER', & run. This will give the following instructions on screen.

Run Type in N
Then Load Menu, Load A File, & Run
Load Menu, Load Another File, Repeat
Repeat Until all Files Loaded
Load Checker & Type Y
Have You Wiped Before?

When running, 'Checker' will show a blank screen until processing is complete. This will take approximately three minutes and then show on the screen:

Ready
Load "Menu", &

Press Return. Run the menu, then load first file, then reset computer by pressing RUN/STOP & RESTORE (or by using a Reset cartridge) Do not switch the computer off & on.

Reload the menu, run the next program, then reset as before and continue until all files have been loaded. Once all the programs on the disk have been run, reload CHECKER, run, and type in Y to the question.

Options is given to output to printer as well as screen. The result of the program is a list of free memory positions available for insertion of the program Returmer. You can select the desired position to place Returmer (if no space available it will tell you so, i.e. Returmer cannot be accommodated on that disk.)



To change the position of returmer in memory

Load 'EDIT', & run

The screen will ask for the new start address. You can enter it in Hex or Decimal numbers (the Checker program provides location values in decimal).

Once entered, press return.

The screen will then ask you to state which file you want to be automatically loaded when you press the RESTORE key. If not MENU, then enter to your own requirements - press Return.

The screen will then ask what name you wish to give the Returmer program. Put your own name in. The screen will then ask "Are You Sure?"

Returner



- If not, type N, if you are, type Y. N returns you to the beginning of the screen, and you will have to re-input memory location etc. When you type Y, it will save the newly-named program to disk.

Starting up your computer

When you start up your computer, you initially load and run the program you named and saved above. It will automatically install the RESTORE key function and run the program you asked to be first loaded (see above).

I recommend that it is the menu as pressing the RESTORE key will automatically return you to this program until the computer is switched off.

Note: A good place to put the 'Returner' program in memory is \$20 (\$204), as it is not normally used when using disk. It relates to the tape drive use, so you won't be able to use the tape drive.

If you have a multipart game, Returner can be used to restart the game by asking it to automatically load in the first part of the game as

part of its program (see above). The start program must, however, have a Basic line number at the beginning e.g. 10 SYS4000.

Addition to Returner instructions

It is not recommended to press RESTORE during the use of the disk drive or printer, as the menu may not work properly if you do so.

If you need to press RESTORE while the printer or disk is in use and the menu doesn't work properly, then just press RESTORE again and the menu should work properly.

Technical Books

Printer Books for the 66. An in-depth handbook which covers interfacing, printing graphics, secondary addressing, graphics dumping, scrolling, listing and more. Includes software disk. 260 pages. Only £14.95

Compiler Design & Implementation (C4 & 66C). Learn how to design a program in a compiler, you will also learn to design a language used to your specific problem & write a corresponding compiler. Assembly & disassembly included. Book & software disk. 290 pages. Only £14.95

Science & Engineering for the 66. An introduction to using the 66 in scientific applications. Describes variable types, computer, float accuracy, computers in science, linear & nonlinear regression, 10⁴ square distribution, Fourier analysis, matrix calculations. Book & disk. 260 pages. Only £14.95

Selected 128 Books

128 Quick Reference Guide. Conveniently lists all of the 128 commands, functions, etc. 66C supported two-stage locations, input/output instructions, program statements, graphics commands & input/output, also graphics & more for all 128 users. Only £4.95

Peaks & Pikes. A collection of useful peaks and pikes for the 128 and their own 66C & software disk. 260 pages. £10.95

Tables & Tips. A treasure trove of programming code including data protection, graphics, sound, interrupt chips. Book & software disk. 500 pages £10.95

BASED P & Hints. A very comprehensive guide to the 128. 66C user-128 pages in the largest 128 book. £10.95 (software disk £14.95)

66C BASIC Training Guide. A self-tutorial guide to programming in BASIC P & H. Full examples and demonstrations. Book & software disk. 260 pages £10.95

GEOS Books

GEOS Inside & Out

The most complete guide to effectively using GEOS. GEOS Inside & Out gives the beginner a gentle introduction to operating GEOS. Later chapters explain the user with GEOPaint & GEOWrite. One chapter is dedicated to practical uses, other topics included are tables & tips, creating custom references for your programs, GEOS file names, etc. Pictoriser, a utility for converting your programs to the GEOS format. Includes optional disk with the book listings, version 1.3. 260 pages. Only £14.95

GEOS Tricks & Tips

A guide to using GEOS. Use the knowledge of an experienced user, over 60 expert hints are included. From setting up documents in GEOWrite to table creation, printing tips, programming your own word messages and much more. Programs included & supported on disk are: Converter, transfer documents into dBaseIII & dBaseIV, a text editor & calculator. Book & software disk. over 200 pages. £14.95

Official 128 Guide

controllers, official programmers guide to the 128

- Indispensable reference guide for the 128
- Basic instructions including:
- The new BASIC P - explore new commands
- Graphics - utilizing the 128's graphics power
- Sound & Music - getting notes out of the 128
- Machine Language - explained in detail
- Operating System - memory management
- Binary files & memory maps
- Input/output guide - peripheral control
- Chips - Possible for hardware
- Over 500 pages. £14.95

128 Repair Guide

Problems with your 66C, are you an experienced or inexperienced user who wants to repair your computer?

- Troubleshooting guide by Howard Pegg
- Simple test finding on the 128
- Subsystems described in detail
- Using test equipment for fault finding
- Component descriptions & characteristics
- Guides arranged by module symptoms
- Preventive maintenance advice
- Easily find the fault & repair it yourself
- Large list size book, full of diagrams, £14.95
- New Troubleshooting Guide 66, £14.95

Anatomy of the 128

Inside your 66C in great detail

- Look deep into the heart of the 128
- Programming Menu & Sound
- Programming graphics modes
- Using the 66C's interrupt chip
- CPU operation and local ROM
- Assembly language programming
- Full commented ROM listing
- Programming the ports
- Optional disk - contains book programs £4.95
- Over 470 pages full of information, £9.95

Anatomy of the 1571

The essential reference guide for all 66C 1571 owners

- For the Beginner & advanced user
- Fundamentals for disk drive operation
- Applying disk drive commands
- Creating libraries & frequent use
- Using your 1571 under 66C BASIC
- Input devices & outputs
- Working with foreign disk formats
- The 66C 1571 system
- Finding programs in the 66C buffer
- The 66C 1571 and 66C software
- Memory disk drive functions
- Many utilities including disk & file copy
- Fully commented ROM listing
- 66C1571 assembly layout
- 66C1571 disk drive module
- Optional disk - contains book programs £4.95
- Over 270 pages, £9.95

1541 Repair Handbook

The book 1541 users have been waiting for

- The 1541 repair & maintenance handbook
- A complete guide to setting for your disk drive
- Topics covered include:
- Maintenance techniques & intervals
- Drive motor speed adjustment
- Testing for correct operation
- Adjusting the read/write head
- Translating to digital electronics
- Complete description of the electronics
- Detailed diagrams on most parts
- Optional disk - contains book programs £4.95
- Large 4th format book with 108 pages £14.95

Hackers Book Pack 64

- Peaks & Pikes 64
- Full of quick peaks & pikes for programmers
- Program tables, sound, use system & more
- Control memory, storage devices, fan charts
- High resolution graphics, keyboard, user port
- (Basic), extensions and machine language
- Over 400 pages packed with information
- Table & Tip 64
- A collection of routines & information
- Includes information including:
- Graphics, colour, scrolling, 3D lines
- Addressed BASIC, Math, 66C1571
- Interfacing & expansion options
- Disk management & sorting, plus much more
- Over 270 pages

• Hackers pack 64 including book disks £14.95

Selected 64 Books

Anatomy of the 66C. Packed with information on the 66C & its peripherals. Chapters include graphics, sound, interfacing, multi-BASIC, assembly language and ROM listings. 260 pages. £4.95

Anatomy of the 1541. Removes the mysteries out of your drive, learn with the step-by-step a program how to write files, use the 1541's disk listing, includes disk & file control, directory mode, file protocol, disk monitor, basic disk & file register. Complete commented ROM listing. Over 320 pages. £4.95

Inside the year 66. Make your 66 work, many essential ideas are given with the software to help you use year 66. Written for the novice, some of the projects covered include drive window searching, register mode & escape code files. Over 200 pages. £4.95

Graphics Book 64. A straight forward book that teaches you how to program, use and design graphics. Create 640 pictures, new character sets, letters & multi colour pictures, apply design & movement. 50 graphics, animation, 66C control and screen memory management & more. Over 300 pages. £4.95

The Official 64 Programmers Guide. A complete book for the 64, published by David Lewis. Topics covered include sound, multi-graphics, word arts, all ports & interfaces, electronic diagrams, Commands, ASCII codes, files & Pikes locations, memory maps. 64/64

Optional disks available for selected books only £4.95 per disk

NEW - Commodore 64 Graphic Software

Clip Art Disks

1000 Clip Art pictures

- 1000 Pictures per disk, 10000 words
- Pictures photo-realistic with every detail
- Disk 1 Variety
- Disk 2 Christmas goodies
- Disk 3 Nature life & scenes
- Disk 4 Animals
- Disk 5 Human figures
- Disk 6 More general pictures
- Disk 7 Sport
- Disk 8 Buildings
- Disk 9 People & faces
- Disk 10 More animals
- Disk 11 Vegetables & Plants, grass, trees
- Disk 12 People's life, more people
- Easily transferred to other disk drives
- Complete disk pack costs 1 to 10 £20.00

**Ideal for Label Wizard,
PrintShop, PaperClip
Publisher & Award Maker
£3.95 each**

Disk Art

Ready-to-use pictures for SDD2

These are not ClipArt but large pictures,
ready to use

- Art 1 - Buildings, paintings, weather general
- Art 2 - US map, life style, buildings, life details
- Art 3 - Buildings, weather, life, flowers, life
- Art 4 - Trees, garden, shapes, birds
- Art 5 - Make a face, flowers, humans, faces
- Art 6 - Animals, dogs, cats, space creatures
- Art 7 - Sports, baby stuff, animals, birds
- Art 8 - Cities, landscapes, computer human
- Art 9 - US life, sports, landscapes, real activities
- Art 10 - Churches, old & old land & houses

- Future - Pre-designed, ready to use home
- Music - Create music sheets

- Created by professional graphic artists
- Designed for SDDPaint & Write, 128 or 64
- Works on 40 & 80 columns, mono or colour
- Disk Art labels Plus disk 1 to 10 £20.00

**High Quality picture disks
£7.49 each**

64 Graphic Software

Newsroom 64 Clip Art disks 1 to 3. High quality
pictures ready to use with Newsroom. 600
pictures per disk. Each Pack £14.95

Certificate Maker. Create certificates for the
Award Maker and present your Certificate
pages £10.95

Certificate Maker Library (Disk 1). Add new
layouts & pictures to Certificate Maker. £10.95

Buttons & Badge Maker. A badge making
factory package. Mark designs included £10.95

SlideShow Creator. Super Slideshow slide
show creator, editor & movie show for your
slideShow. Used for Action Replay, Super, Power
Slide. Only £20.00

Graphic Label Wizard

Great new graphics label printing utility for
the C64

- Print labels with both text & graphics
- Easy to use menus with pop up windows
- Use PrintShop/Fontmaster ClipArt pictures
- Other graphics display - sets up a picture
- Quickly design, format & print your label
- One graphics plus 4 lines of text per label
- Save labels to disk for later editing or printing
- Print multiple copies of your favourite label
- Print 1/2 dot or labels across
- B&W or 16 extra pictures

**A flexible printing tool
£19.95**

Newsroom 64

A disk too publisher for the 64

- Design, produce & print newsletters, posters
- Extract from disk clipart pictures
- Word processor with 4 fonts in 4 sizes
- Photo Lab, colour and text elegant pictures
- Banner, photo page, message features
- Copy Paste, word processor with 4 fonts
- Layout Editor, quickly design your pages
- 8 disk package with an 88 page manual

**A dynamic program for
journalists of all ages
£24.95**

Award Maker Plus 64

If it's worth hanging, then it's worth
celebrating with an award they'll keep

- Create beautiful awards with your 64 & printer
- Ideal for any occasion, design your own
- Easy to use - four steps to create an award
- Extract text from many sources
- Select colour to improve printing
- Choose a graphic from the large picture library
- Type in a message, date & add your signature
- Print out to your domestic printer
- Save names & print awards for future classes
- Use PhotoShop/ClipArt pictures for borders
- FREE 8000 embedded press art text

**Create a keepsake they'll be
proud to display
£24.95**

Video Title Shop 64

Become your own Movie Director

- Use your C64 & video to film your home movie
- Create an endless array of title effects
- Speed fading, down or across
- Picture only or audio only options, freeze a screen
- Combine text & graphics, effect & more

**Polish your home movies
£19.95**

Paper Clip Publisher

The most powerful Desktop Publisher for
the Commodore 64

- Produce anything which uses words & pictures
- Newsletters, reports, presentations, flyers

- It's easy to use -
- When you see it - when you get things
- Put direct access to get you results fast
- Icons for commonly used tools
- Drag-in text editor to prepare your final text
- Drag-in graphics editor to touch up effects
- Mouse or joystick control

- It's easy powerful -
- Produce multi-page documents
- Design boxes for text or graphics
- Move boxes anywhere on the page
- Better boxes & automatically format text
- Flow and around artwork
- Import graphics from PrintShop, Newsroom
- Use ClipArt pictures
- Import text from most word processors
- Set columns, character spacing & margins
- Great instant crop window & fit picture

- Create Graphics modes -
- PhotoPaint, paint, colour
- Zoom (post editor), pencil, erase
- Graphics importer

- Textile features & fonts -
- Bold, light, superscript, italic, mirrored
- Shadowed, text effect, underlined, outlined
- Fonts - Helvetica, Times, Courier, Symbol

**An impressive desktop
publisher for home and
professional use
£24.95**

Solutions Unlimited

Photo Finish

- Four-year Quality - available on your printer
- Optimize your titles graphics producing -
- Working with four-step film print resolution
- Compatible with most framing packages
- Print 50000 pictures of 160
- Photo Finish for the C64 £24.95

Icon Factory

- Derived graphics from one format to another
- Images, PrintShop, Fontmaster, Quattro
- Drawing PaintBox, PrintShop etc
- Image editor - enlarge, smooth, invert flip
- Crop and cut pictures
- Convert 16-bit pictures to monochrome & text
- Icon Factory for the C64 £24.95

To Order

All products available by mail
order, call now with credit
card details or mail a cheque
or Postal Order to the address
below.

Prices include Postage & VAT.

Allow 14 days for delivery

ORIGINAL BASIC SOURCE LISTING									
100	100	100	100	100	100	100	100	100	100
200	200	200	200	200	200	200	200	200	200
300	300	300	300	300	300	300	300	300	300
400	400	400	400	400	400	400	400	400	400
500	500	500	500	500	500	500	500	500	500
600	600	600	600	600	600	600	600	600	600
700	700	700	700	700	700	700	700	700	700
800	800	800	800	800	800	800	800	800	800
900	900	900	900	900	900	900	900	900	900
1000	1000	1000	1000	1000	1000	1000	1000	1000	1000
1100	1100	1100	1100	1100	1100	1100	1100	1100	1100
1200	1200	1200	1200	1200	1200	1200	1200	1200	1200
1300	1300	1300	1300	1300	1300	1300	1300	1300	1300
1400	1400	1400	1400	1400	1400	1400	1400	1400	1400
1500	1500	1500	1500	1500	1500	1500	1500	1500	1500
1600	1600	1600	1600	1600	1600	1600	1600	1600	1600
1700	1700	1700	1700	1700	1700	1700	1700	1700	1700
1800	1800	1800	1800	1800	1800	1800	1800	1800	1800
1900	1900	1900	1900	1900	1900	1900	1900	1900	1900
2000	2000	2000	2000	2000	2000	2000	2000	2000	2000

There is also a screen option for checking the disk directory, and to make sure that any files have been successfully saved.

The reports given by the state generator and the flow and data cross-reference options can be saved to disk alongside the finished program. There is no direct hard copy option, but a safety file is included which will read a stored report to the printer for future reference.

Arranged Crush

The final pre-compilation routine is the Compressor, which reduces the program to its absolute minimum size. As each line is scanned, the routine removes all REMs and spaces but the compression goes beyond this. Lines are combined in sequence in as many commands onto a line as possible.

After compression, a printout of the program may contain lines which physically span three or more screen lines. This is possible because the Compressor works with raw program code so that a command such as PRINT only occupies one byte rather than five. A line could therefore contain as many as 38 PRINT commands, which would expand to 228 characters and colons or five screen lines! This would make editing impossible; so the program must be thoroughly debugged before this operation is performed.

Although this optimises the program's memory usage, difficulties were experienced when using the compiler so I eventually decided that compression was probably better used on programs which wouldn't be improved by compilation.

Full Speed

CA-Comp is one of the best compilers that I've ever seen. It is a full-facility, four pass compiler which can cope with all of the commands in the C64's Basic library. The program also

accepts all types of variables and arrays up to two dimensions. Three-dimensional arrays can be compiled, but only after the program has been pre-processed by a special utility which is included on the disk.

A compiler converts Basic code into a special form of machine code which runs up to 48 times faster than the original program. When a Basic program runs, the operating system has to scan each line in turn, turn each command into an action, and create and modify all of the variables it encounters.

A compiler scans through the Basic program and converts all of the commands and their parameters before the program is stored on disk. The conversion also means that sufficient space can be made for all of the variables which are to be used. The finished program is therefore midway between Basic and pure code.

The initial compilation can take a long time to produce, because the full program is scanned and re-scanned four times before the final program is produced. During these passes, the compiler produces and creates temporary work files which add disk access time onto the process. A long program can take up to half an hour to prepare, but the results are worth waiting for.

Warily, the producers insist that the master disk should be used as little as possible, and a special back-up routine is included which re-scans the compiler and its associated utilities onto separate disks. Unfortunately, the back-up is not the most efficient of utilities, and each sub-program is transferred individually, regardless of whether it occupies one block or 20. The result is a constant swapping of disks, which makes the whole process very laborious and unnecessarily long.

Once the copies have been made, the compilation process is automatic, and the program to be processed is placed on the same disk as the compiled compiler. This can cause problems with disk space, but this problem has been anticipated by offering suitable error message generation and options to erase programs which will no longer be required for the final compilation — even the original program can be erased once compilation has progressed beyond the first pass.

The limitations of the compiler are surprisingly few and petty. Arrays must be finitely dimensioned, which means that DIM A(21) is a legal statement but DIM A(X) would be

quoted. If an array slips through to the compiler, which lacks an explicit dimension, the program interrupts compilation to ask for a suitable array size before continuing through the rest of the program. This may limit the efficiency of memory usage, but few programs would suffer appreciably by this insistence on explicit dimensioning.

The only commands that the compiler rejects are LIST, CONT, SAVE and RUN when it has a line number parameter. LIST is excluded because there is nothing to list in a compiled program, CONT is an acceptable omission and most people could live without commands such as RUN#0 when a CLR: RESTORE: GOTO#0 could be substituted instead. SAVE is harder to live without but this could be remedied by including an STS in the original program to a suitable machine code patch.

One worrying problem which revealed itself under testing was that the maximum string length is 254 characters, instead of Basic's 255. This must surely be an oversight on the part of the programmers because it is not documented in the manual. Such errors are slightly disconcerting because they only show up at runtime, and I'd rather not sit with my fingers crossed while a program compiles hoping that another unforeseen error occurs.

All compiled programs are executed with the RUN/STOP key disabled. In most cases this is unimportant but it may be necessary to allow such an interruption as a special form of REM statement has been devised to allow the key to be enabled and disabled.

Error Tracing

If errors occur when the compiled program is being tested, it can be difficult to relate the problem back to the original program. An error which stops the execution of the program will generate a message which is similar to Basic's equivalent message, but instead of a line number a memory location is displayed.

A utility on the disk can be given this value, and it will then automatically cross-reference this number to the original program to assist debugging. Of course, such an error should rarely occur because any sensible programmer would have tested and

debugged the original program before compilation, wouldn't they?

Another utility which can aid debugging was actually devised for hybrid programs which require parameters to be accessed by a machine code patch. Variables and arrays are stored differently in the compiled program so any associated code routines have to be modified. This works in favour of routines which access a current variable value, because the memory location of variable data is fixed in a compiled program. **REPORT** is a utility which scans the original program and prints out a table of variable locations to ease the problem of conversion.

Dotty Protection

Why, in this day and age of colour photocopiers and cameras, do software houses insist on these ridiculous colour charts? I don't mind protection systems, but I object to this form because I am red/green colour blind and the companies always seem to insist on using red, green and two other

colours on these charts. Why black, white, red and yellow can't be chosen defies me.

In this case the problem is increased because each of the four programs is individually protected. In the end, I got so annoyed that I resorted to backing up the programs using a cartridge copier - so much for protection!

Despite this, the package maintains a very high standard throughout all four sections. Maybe if the space under the ROMs had been used, the toolkit could have been more comprehensive in the manual, but no computer is perfect and this compiler is comparable, though slightly inferior, to Supersoft's *Altos*. What this has to offer which *Altos* doesn't is the other three packages which make this package stand out against the opposition.

Toolkit Commands

CHANGE
DELETE
DUMP
FIND

INFO
MERGE
QUIT
RENUMBER
SIZE
TYPE

Analyzer Options

Flow cross-reference
Data cross-reference
Examine disk directory
Deadwood analysis
Create header file
Generate stats
Return to Basic

Compressor Functions

Removes spaces and REMS
Compress program lines

Ox-Comp Facilities

Majority of Basic 2.0 commands
Four pass operation
Error checking
Location of variables printed

AT LAST A POOLS PROGRAM THAT DELIVERS THE GOODS!!

POOLSBUSTER64 IS HERE!

THE PROGRAM THAT GAVE HUNDREDS OF DIVIDENDS TO ATARI ST USERS, IS NOW AVAILABLE FOR THE COMMODORE 64. POOLSBUSTER64 IS QUITE SIMPLY THE MOST ADVANCED POOLS PROGRAM AVAILABLE TODAY. LOOK AT THESE HIGH-SCORING FEATURES: 1) POOLSBUSTER IS GUARANTEED THAT'S RIGHT-WE'RE SO CONFIDENT THAT YOU'LL WIN WITH POOLSBUSTER THAT WE PROMISE TO REFUND THE PURCHASE PRICE IF YOU HAVEN'T WON SOMETHING WITHIN ONE YEAR OF THE DATE OF PURCHASE! 2) IT'S THE STATE-OF-THE-ART POOLS PROGRAM. IT USES AN ARTIFICIAL INTELLIGENCE (A.I.) SYSTEM TO FINE TUNE ITS PREDICTIONS EACH TIME YOU ENTER A SET OF SOCCER RESULTS. IT ACTUALLY LEARNS FROM THE RESULTS IT GETS WRONG. 3) IT KNOWS THE SCORE! POOLSBUSTER CONTAINS A MASSIVE DATABASE OF SOCCER STATISTICS WITH DETAILS OF OVER 1000 PAST MATCHES. 4) IT'S EASY TO USE. POOLSBUSTER IS FULLY JOYSTICK/MOUSE DRIVEN - THERE'S NO NEED TO USE THE KEYBOARD AT ALL. 5) IT'S VERSATILE. POOLSBUSTER COMES WITH ALL THE U.K. SOCCER LEAGUES YOU'RE LIKELY TO NEED, INCLUDING QM VAUXHALL, NORTHERN PREMIER, BEAZER & HPS LEAGUES. AND YOU CAN ADD ANY OVERSEAS LEAGUES AS YOU WISH. 6) YOUR FORTUNE IN THE STARS! POOLSBUSTER64 ALSO INCLUDES THE UNIQUE MAGIC PREDICTION PROGRAM. THIS FORECASTS LINES OF 16 POSSIBLE DRAWS ACCORDING TO THE ASTROLOGICAL POWER NUMBERS FOR YOUR NAME, DATE OF BIRTH & POOLS DATE. POOLSBUSTER64 IS AVAILABLE ON 5.25" DISK OR CASSETTE.

**POOLSBUSTER64 COSTS ONLY £40.1 SO HURRY
ORDER ONE TODAY & WIN THE POOLS TOMORROW!**

VERSION 5.25 - 10000 BITS/SECS. BOX OF THREE COMMODORE 64 DISKES. EITHER 10 OR 15 MINS. D.C. BIAS. THE BEST MEDIA FOR YOUR PROGRAMS (also FOR 16 SPECIAL DISKS IF YOU WANT to use Data Bases, Journals, Photos, etc.) ORDER NOW. SEND ONE POUND TO YOUR CARD NO. - 100000 DATE TO: (SOUTHERN SOFTWARES, 8PT FORD, SOUTHERN BUSINESS CENTRE, UNIT 10, ALEXANDRIA ROAD, 100 BATTERSEA PARK ROAD, LONDON SW8 5NN. TEL: 01/978-2280. 24HR: 01/738-8400. FAX: 01/622-1063

```
10 PRINT "PROG"
20 PRINT "RUN IT"
```

```
40 GOTO 80
```

```
50 RUN
```

```
60 LINE MISSING! HELP!!
```

Line Input

*Improve your 64's input
with this handy utility*

There are times when the standard Basic keyword "INPUT" on the C64 is a bit of a pain. In the event that a comma (",") needs to be input as part of a string "INPUT"

cannot do it, because the comma is interpreted as being a delimiter between different bits of data which are supposed to be assigned to separate variables.

If you wanted to enter such a string then it has to be enclosed by quotes so that "INPUT" will read in every character. This looks silly on the screen, and is very confusing for users

of the program who are not familiar with the technique - they can't understand why some text has to be enclosed in quotes in order to enter strings including commas, whereas other forms of input such as numeric data do not. It's even more confusing when the program later requires that the user use the comma to separate input into different fields!

The other slightly aggravating thing about "INPUT" is the fact that the "?" prompt is always printed on the screen prior to "INPUT", waiting for keyboard entry. This is sometimes a nuisance, since the message which asks for the input on the part of the user is not always of a kind where a question mark at the end is relevant - that is, it's not a question. Add to this the need to follow it with quotes - and then rub this out and type quotes again so you can use the cursor move keys to edit the line - just so a comma can be included where required in the input string, and straight away we already have two irrelevant characters on the screen before anything is actually typed in.

What's more, you always forget to use the opening quotes at the start, which itself suggests that it doesn't feel natural to have to do this. The situation is worse when inputting from an external channel - here not only the comma but also the colon and semi-colon are interpreted as carriage returns, or the end of the string, whereas in actual fact it may be nothing of the sort. Again, enclosing the string in quotes before it is saved to the external device prevents this, but as inputting in back the quotes will have disappeared. In either case, whether inputting from the keyboard or an external device, quotes cannot be included as part of the text. But why not, you may ask?

"GET" will input anything from the keyboard, and a string comprising all printable characters can be built up using "GET". But the Basic coding required to interactively edit this string at the same time rapidly becomes unwieldy and slow. Also, the input is not automatically echoed to the screen, so this must be done in Basic. Then's no flashing cursor either.

A cursor can be forced, but it has the peculiar behaviour of leaving some characters in inverse video as soon as the cursor move keys are used. "GET" can be used to input text from an external device, but using Basic commands that retrieve one character at a time and build it into a string can be time-consuming to say the least. It's frustrating when all you want is input some text of an 'unusual' format.

If you're a machine code programmer of any skill, then this can be got around, except that you may find yourself having to re-develop the coding to suit the requirements of different programs. But if you want

to stay in the Basic environment, then while "GET" and "Input" have their place (and are very good at what they do), what is really needed is a new Basic keyword, one that will input strings of all printable characters.

Well here it is. Inspired, as it happens, by the IBM PC BASIC A's action keyword of the same name the ability is called "LINE INPUT" and is in the form of a machine code routine stored at SCAB0 to SCB0D high up in the free RAM area above the 64K BASIC interpreter. In actual fact, it is not a real keyword at all, but a "SYS" call which claims by pretending to be an interpreter routine.

This is necessary because of the sort of information that the routine needs in order to carry out its task. The "USR" function only allows one numeric value to be passed on to a machine code routine, which then can only return one other value to a variable. The temporary register storage for "SYS" calls are rather long-winded to use, and in any case none of these are any use because "LINE INPUT" must return a string in a string variable.

The simple answer is for the routine, "LINE", to get these parameters direct from the current Basic line being processed using the subroutines "CHRGET" - which retrieves the next BASIC byte into the accumulator, and also incidentally is the thing that skips spaces not in quotes - in exactly the same way as all the interpreter routines get their information. This makes "LINE INPUT" very easy to program in Basic, because if you excuse the presence of the word "SYS" at the front it will appear in the listing, and in operation, like a valid Basic keyword. Furthermore, if you make a numeric variable equal to the "SYS" call address and name this variable "LINE", the subroutines will be complete. The system then is as follows:

```
SYS < address > INPUT
( if < channel No.> < string variable >
```

Note the full stop between < address > and INPUT (if < channel No.>). This is very important as it ensures that the following valid BASIC keyword, "INPUT" or "INPUT# ", is crunched down to its proper BASIC token when the BASIC line which contains it is entered. Neither of these will be executed by the interpreter in the normal way. However, they exist here exclusively

for use by "LINE". In fact, the processing activity of "LINE" will make whichever of these two tokens inaccessible to the interpreter.

The keyword for "INPUT" or "INPUT#" is included because "LINE" has to sortie within itself to the appropriate processing for either inputting from the keyboard and screen, or from an external device. The keyword is tokenised for three reasons: 1. to save memory space, although this will be negligible; 2. because it's a professional way of doing it; 3. since "LINE" uses it to find out which form of input is required, it simplifies things greatly if only one byte can be examined instead of a string of characters. This token, and the remaining parameters, are read by "LINE" direct from the Basic text using "CHRGET".

A number of subroutines are used by "LINE" to read the parameters. They themselves get the Basic bytes using "CHRGET". One of these is a contrived subroutine called "GETVAL" located at SCBA7 (\$2113 decimal), and looks like this:

```
READ EQU $ADDA
FIX EQU $B7F7
ORG SCBA7
GETVAL JSR READ
JMP FIX
```

"READ" is part of a Basic interpreter routine which picks up a numeric value from the Basic line currently being processed. The value may be written as a string of decimal numbers, or be in the form of a numerical variable. Whichever, "READ" converts it into floating point format and stores in Floating Point Accumulator \$1.

"FIX" is the familiar 'fix to fix' interpreter routine. The contents of FAC - 1 are converted into a 16-bit integer and stored in zero page locations \$14-\$15 hex.

Therefore "GETVAL" performs the action of reading a numeric value as a variable's contents from a Basic line and making same available in handy double-byte integer format in \$14-\$15 for a machine code to use. If "GETVAL" is handled properly, it helps in understanding how "LINE" operates, that the C64 Basic interpreter follows a special convention that every interpreter routine, including those used by "GETVAL", expects the first Basic byte that it has to deal with to already be in the CPU's accumulator. Or, to put it another



way, every Basic interpreter routine calls "CHRGET" to get the next byte of Basic text into the accumulator before it exits. Also "GETVAL" is a little bit limited, only positive numbers in the range 0-65535 are allowable otherwise an "ILLEGAL QUANTITY" error is generated. Invalidly entered numbers produce "SYNTAX ERROR".

On loading "LINE" with the "SYS" command, true to the convention, the interpreter runs "CHRGET" to load the accumulator with the next byte of Basic text. Unfortunately this doesn't arrive intact when "LINE" takes control, because the action of successive "SYS" loads the accumulator with the contents of the temporary storage for A reg. located at \$060C (260 decimal). So the accumulator has to be refreshed with a call to "CHROUT", a letter part of "CHRGET" which gets the same BASIC byte again. This byte must be a full stop (".") or EOL, if it is not a jump is made to \$AF08 which prints "SYNTAX ERROR IN LINE <n>", and terminates the Basic program. The byte following this, retrieved by "CHRGET", must either be a token for "INPUT" or "INPUT #". If it is neither of these again a jump is made to "SYNTAX ERROR".

Observing at this point the routine "LINE" deviates. The bulk of the routine comprises two separate processes, the one chosen depends on whether the input is from the keyboard and screen, or whether it is from an external device.

LINE INPUT from keyboard and screen -

Source:

SYS < address> INPUT < string variable>

Example:

```
100 LINE = $0900
110 SYS LINE INPUT A$
```

Input from keyboard and screen is carried out in three stages. Firstly the screen RAM address for the start of

the input is found by loading the screen position with the kernel routine "PLOT", and calculating the screen address by the rather crude method of adding 48 to the screen base of 1824 ("Y" (row count) number of lines, "X" (column count) is added to the result. I did originally write a proper multiplication routine, but this required so much code that it wasn't really worth it. The calculated address is stored in free zero page locations as a pointer.

All you see on the screen is a flashing cursor with an "??" prompt. Now a kernel routine called "CHREN" handles all the character getting and printing with full editing support and proper flashing cursor. In use it is identical to using the Basic editor in direct mode. It has the same limitations - if quotes are typed then cursor move keys cause onto the screen as characters, but rubbing out quotes and re-laying stops this. "LINE" discovers the length of the line. This is done by searching backwards from the maximum string length position, the default of which is 80, looking for a non-space character. On finding a non-space character, its position ahead of the start of input becomes the new string length.

The screen line, from the start address up to the string length, is copied to a buffer. This is our old friend the cassette header buffer at \$030C. However \$030C is reserved to hold the string length count, so the buffer actually starts at \$0300. A conversion routine is used to translate screen coded characters into CBM ASCII, and the most significant bit is stripped off in case the screen characters were inverse video ("LINE INPUT" doesn't input a line which has just been typed, it could just as easily have been PRINTed instead and the keyboard buffer FORKed with a carriage return).

At the end of this the string is stored in the buffer and its length is stored. At the final stage "LINE" calls interpreter routines "LOCATE", which goes in search of the string variable through the variable storage

area - "LOCATE" gets the variable's name from the Basic line. If it does not exist then it is created. After this the string storage for the variable, if already in existence, is freed (by "FREE") and then a fresh storage area is defined using "STORES" which does so according to the new string length passed to it. We now have a place in the string storage area to which the contents of the buffer can be copied.

It is impossible for characters to be put in this area directly (without being stored in a buffer first) because "LINE" does not yet know how long the string will be until this stage.

In this manner any string variable can be created or updated using "LINE INPUT" is just the same way as it can by any other INPUT function. However some care is required where the variable is one of a dimensional array. Small arrays up to a 'safe' limit of say 10 or 11 elements would be okay, in other words:

```
110 SYS LINE INPUT T$(0)
```

Where 'n' is an index from 0 to 10 in the default provision for arrays up to 11 elements, but otherwise a 'working variable' such as IN\$ for example would have to be used to transfer the input to the element where this is one of a large, dimensional array, e.g.:

```
110 SYS LINE INPUT IN$
120 A$(0) = IN$
```

If this precaution is not observed the process becomes too complex for the simpler than normal string building techniques used and usually crashes the 64.

Whatever the string is stored in the proper place, and the extra does not add perceptively to the processing time, even if a large string array is being filled with text.

Because "LINE" is located in RAM it can be modified. I mentioned that for input from keyboard and screen, it has a default maximum string length of 80 screen characters (two screen lines). It doesn't matter in which column across the screen the start of input occurs, the maximum remains 80 characters. You can change the maximum length by a POKE to suit your own requirements. For example, a disk file name cannot exceed 16 characters in length. If "LINE INPUT" is used to get the file name it can first be modified only to accept

up to a maximum of 16 characters. In the source listing the maximum string length is labeled "MAXLEN" and is at \$C04D (\$3045). Use the following:

```
100 LINE = 51004
```

```
1000 MAXLEN = 16045
1005 POKE MAXLEN, 16
1010 PRINT "ENTER A NAME
FOR THIS FILE".
1020 SYS LINE INPUT IN$
1040 NMB = IN$+" 9/8"
1050 OPEN 2,2, NMB
1060 etc.
```

You can type as many characters as you like for the file name, only the first 16 will be taken any notice of after you press [RETURN].

Be aware though of a couple of limitations of "LINE INPUT". One is the same as that of the normal "INPUT", namely if other characters prevent on the screen come within the range of the maximum string length, it will be assumed that these constitute the end of the string, even if you don't want them. Enough blank space must exist beyond the position where input commences to prevent this.

Also, "LINE INPUT" doesn't know if the start of input occurs on the bottom line of the screen. If the screen scrolls up where text exceeds one screen line, the start address is not adjusted up with it. Also "LINE INPUT" will assume that the "rubidish" beyond the top of the screen RAM area are characters to be input! Consequently "LINE INPUT" should not be used any lower than the 24th row down.

LINE INPUT # from an external device -

Syntax:

```
SYS < address >, INPUT < channel No. >, < string variable >
```

Example:

```
100 LINE = 51004
```

```
1000 OPEN 2, 2, "TEXT"
1010 SYS LINE INPUT 2,AS
1020 etc.
```

This is the only time that "GETVAL" is used, and even then only the least significant byte of the resultant integer is needed!

"GETVAL" retrieves the channel number following "INPUT #" so that

internal routines can route input from the appropriate channel. Inputted characters are copied to the buffer, a process that continues in a loop until one of three conditions are true:

1. The ST variable indicates an EOF signal from a disk drive. On detecting "EOF" inputting is terminated.
2. The character read is found to be the termination character. The default termination character is a carriage return (13), on encountering the termination character inputting is terminated. The termination character is NOT copied to the buffer.
3. The maximum number of bytes have been copied to the buffer. The default maximum number of bytes is 128. On the 128th character being copied to the buffer, inputting is terminated.

On input being terminated, "LINE INPUT #" goes to the final stage of building the string to memory exactly as before. As far as "EOF" is concerned, it will be required that the BASIC program using "LINE INPUT #" should also monitor the "ST" variable. Re-using "LINE INPUT #" and forcing input past EOF causes a rubbish character to be copied to the buffer, overwriting the previous string. This is not a problem if other variables have been made equal to this string.

As before, because the routine exists in RAM, it can be customized to behave in a special way. "CHECKR" is a label in the source listing which represents the point where the character read is tested to see whether it's the termination character or not, it looks like:

```
CHECKR CMP # 50D
```

Where the operand is the value, in this case 13. This can be changed as in the following example:

```
CHECKR = 32005
POKE CHECKR + 1,0
```

Now the routine will stop reading when it encounters a zero byte. Similarly the maximum number of bytes copied to the buffer can be changed:

```
CHEMAX = 52079
POKE CHEMAX + 1, < n >
```

Where < n > is any number up to a limit of 191, because this is the maximum amount of space available in the cassette buffer area.

Even better, "LINE INPUT #" can be made to ignore any termination

character. Following the "CHECKR" location is this:

```
INTERM BEQ PUTLEN
```

which is where a branch is made to the final stage upon encountering the termination character. By using

```
INTERM = 52063
POKE INTERM, 224 : POKE
INTERM + 1, 234
```

This has the effect of overwriting the branch instruction with "NOPs", so the routine cannot exit if a termination character is found. This allows maximum flexibility for inputting data of a "strange" nature - "LINE INPUT" can read in all byte values 0-255. It is possible for example to read machine code into string - supposing you wanted to do such a thing!

Because "LINE INPUT #" is completely self contained, i.e. no part of the conventional interpreter (INPUT #) routines are employed, we are allowed to do something normally quite illegal. The following is possible in direct mode:

```
LINE = 50904
```

```
READY.
OPEN 2, 2, 2, "TEXT"
```

```
READY
SYS LINE INPUT 2,AS
```

```
READY.
145
```

This is incredibly useful for verifying that a Basic routine that you are trying to de-bug has saved data properly or not. It's normally impossible to get at this data in direct mode, since using "INPUT #" results in an illegal direct command error. You can go on moving the cursor up an displaying and displaying successive fields of data. This even works with numeric data saved using "PRINT #", since these are written as decimal strings.

If you have an assembler you can enter the source listing, and if needs must the ORG statement can be changed to relocate the routine anywhere you like, but make a note of the new label locations if you then want to customise it with POKEs. It's unlikely that you will need to use "LINE INPUT #" very often, but when you do, you'll be glad it's there.

A Flow of Ideas

It's often necessary to view a directory from within a program, but from Basic this can be very difficult. There is a way, and this method can reveal more information than may at first be obvious.

Before a disk directory program can be written, it is essential to know what data the drive makes available. Using the following short program on the directory in Table 1 gives all of the information which is displayed in Table 2.

```
10 OPEN "A:", I, "R", 8, 0
20 GET #I, DIR$(I) : PRINT DIR$(I)
30 IF DIR$(I) = ":", GOTO 100
40 IF DIR$(I) = ":", GOTO 100
50 IF DIR$(I) = ":", GOTO 100
```

The enclosed data includes the Block Allocation Map (BAM) which lies from BC000-BCXXX, as well as the directory constant itself (BCXXX-SCXXX). Using GET #, each byte can be read and used to form quite a powerful source of disk information.

From this constant, the aims of the program can be derived. What we will create is a flow diagram for a program which lists each directory entry, the file type, blocks used and the track and sector (the specific block location) relating to each file. As the program runs, it will keep track of the total number of programs on the disk for a general run-up screen, which is to be displayed after the individual program detail screens.

The program can also keep a tally of the total number of blocks used, and this value can then be compared with the number derived from the BAM to ensure that the disk has been validated correctly. Finally, a detailed map of the number of free sectors (blocks) on each track can be displayed using colour to differentiate between tracks which have not been used at all, and those which have lost a few sectors to file storage.

The first duty of the program is to set the screen colours and then initialise the disk under examination. This means opening command channel 15, and keeping it open for checks on disk errors throughout the program.

Next, the directory file is opened for a sequential read operation. After the drive loads the first sector into its

*How easy is it to
incorporate a disk directory
reader in a program?*
By Norman Doyle



internal buffer, variables are initialised and dimensioned ready for the main program to begin. The reading of the BAM takes a little time, so a comforting message is displayed to assure the user that this is a normal occurrence.

Exploded BAM

The layout of the BAM can be seen from Table 3. There is no single byte pair responsible for recording the number of free bytes on the disk, so this value has to be derived from the data at the beginning of the BAM. The first two bytes (the track and sector index) are not loaded when the directory is read as a sequential file, and the next two bytes are irrelevant, but some of the next group of bytes are essential to calculating the number of free memory blocks on the disk.

The information is stored in a specific map, with the bytes grouped into clusters of four. The first byte is the total number of free blocks on a particular track, and the three bytes that follow can be used to calculate which particular sectors these are. In this case, the only byte of interest is the total number of free blocks, so that is read and added to a running total while the others are discarded.

Apart from keeping a record of the total number of free blocks, each

track's details must be stored in an array if the aim of producing a tracks map is to be fulfilled. There are 35 tracks on a normal Commodore disk, and it can be seen from Table 2 that there are, in fact, 35 groups of four bytes in the BAM. A loop must read each of the groups, storing the first byte in an array and adding its value to a grand total of free blocks. After this is done, the next three bytes can be ignored.

At the end of the BAM, the disk name and ID numbers can be found. This must be read and stored in a suitable form for display at the head of each screen page. First of all, the string is initialised to produce a label indicating that what follows is the disk's title and, to comply with Commodore's convention, the EYS ON character is added to display the disk header in reversed characters. A second convention is to place the disk name in quotes. This is not catered for from reading the directory file, so the opening quote is added at this point.

The disk name is allocated a space of 16 characters in the BAM sector of the directory. This means that by simply reading the 16 character group and adding it to the title string, any disk name can be catered for without any complicated checks. The resultant string is then completed with a closing quote mark.

To complete the screen page header, the next 15 bytes are added to the string. This actually reads more than enough characters to cater for the ID and the disk type descriptor (always 240 on a normally formatted disk, but is extended to cater for non-standard formatting using four character IDs and other such tricks. The difficulty here is that some of the bytes will be stored as 'null' values, which is the reason why string variables are being used in the program instead of numerical ones to avoid generating errors.

Because the bytes are stored as ASCII values, the conversion routine must look for these nulls and convert them to CHR\$(0) to avoid similar errors when the bytes are converted for use. This happens quite a lot, and a sub routine is the ideal answer. Once the disk name has been included in

a string, it is printed and, because this string contains the clear screen symbol, the same acts as a title for the screen page.

Before reading in the directory entries, the column heads must be displayed. The convention that has been devised for this routine is that the program name will be followed by the file type, block count and the track and sector values for the first program block.

This information is not stored in the correct order as it must be read, stored and sorted out in a print statement. The layout of a single directory entry can be seen in Table 4. A sub-routine deals with the actual reading of the individual entries, and this will be examined later.

As each directory entry is dealt with, a counter is incremented so that a check can be made for a full screen page (20 entries). When this number is reached, a 'Press Any Key' message is printed and a keypress detection loop initiated.

Directories come in all shapes and sizes, so the programs cannot work on a simple loop to read in the entries. A way is needed to indicate when the directory is complete. This is done by reading the system variable, \$T. When this has a value of zero, work is still in progress, but if it has a non-zero value one of two things has occurred.

It may just be that the directory has been read in completely, so it could be that an error has occurred. Before closing the file, the routine checks the error channel. If an error is detected, the program will halt and the message is printed; if there is no error the program continues to the next stage.

Under a screen page title of 'Vital Statistics', the details of the number of programs, and blocks used are printed, suitably labelled. Both are derived from the individual directory entries and supplied via the 'read entry' sub-routine.

The free blocks' value has been calculated from the BAM record, but before this is printed we can use this value and the 'blocks used' value to check for a correctly formatted disk. If the number of blocks used is subtracted from the total number of blocks free on a newly formatted disk (64K), the result should equal the BAM-derived free block count.

Any inequality means that the BAM is faulty, and this is flagged by a suitable message. A word of warning - inequalities could be the result of

RLL, or VSR, files being used on the disk, so ensure that this is not the case before validating, or you could lose valuable data!

After the free blocks have been displayed, a table is drawn up with the tracks listed above their unused sector values. At each stage, these values are checked against the sector capacities for each track. If none of the sectors have been used, the value is printed in brown, but any tracks which contain file data are highlighted by using light blue characters.

This program can easily be extended and improved to give a full BAM map, or reduced to produce a disk-to-screen directory printer.

The sub-routine to reach each directory entry is an integral to the running of the main program and, therefore, worth a closer look.

Directory Delivery

The directory read routine takes over immediately after the disk name has been read from the disk. The directory details are provided by a series of null bytes which must be discarded until a 'real' byte value is read in. This method saves time but has one serious drawback when the first file has been deleted and not replaced.

File types are denoted by byte values as follows:

```
129 = RLL
130 = PRG
131 = USR
132 = RLL
```

If the file is 'locked', or protected, these values are increased by 64 to give a range of 193-196. Deleting a file simply results in these values being replaced by a null byte. Using the method of discarding null bytes would mean that the first significant value would be the old track number when a deleted file was encountered.

Consequently, the program includes a check to see if the value read is in excess of 192. If this is not true then it must be a deleted file, so 'DEL' is assigned to the program type and the routine jumps to calculating the read in value as a track number. If the value is valid for a file type, a check is made to test if it is locked or not. Previously it made to denote a locked file by using a reversed 'less than' symbol, which is saved as a string. If the file is unprotected, this string remains as a space.

A second string is derived from the file descriptor value. This is the file type which is calculated directly from the file type value in conjunction with a MATH statement. After the next byte has been read in, the pathway which the DEL file option took converges with the main program so that the track value can be assigned to a string before the sector value is read and similarly stored.

The disk name had 16 bytes assigned to it, and the same is true for the program name which follows the sector byte. A loop reads this in and concatenates a string. If a program name has less than 16 characters, the entry is padded out with shifted space values (160), and this is useful for formatting the screen printer later, so they are not discarded here.

After the name there are nine zero bytes which are read in and discarded, so the next significant bytes denote the number of blocks that the program occupies. This value is stored in low byte/high byte format and a suitable routine is included to reveal the true decimal value. All that remains is to print the information out in the correct screen columns. Deleted file names are shown in orange and existing files in white.

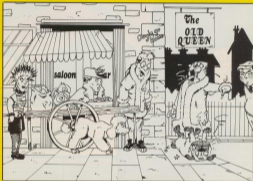
One problem with filenames is that they sometimes contain control characters such as screen clear, reset switch or even colours. To avoid these messing up the beautifully formatted display, location 212 is pointed with a value to fool the computer into thinking that it is in quote mode, so that the reversed character is printed instead of the action which it represents being executed. Once the rest of the information is printed, control is then returned to the main program.

Directory Flow

The flowcharts show the logic of the program contained in the Listings pages. This is rendered as a Basic routine, but the beauty of flowcharting is that the same logic pathways can be applied to machine code or just about any other language that has been invented.

If you decide to investigate directory reading further, you'll also soon realise how useful the charts are for modifying the program. It's far better than moulding through frames of listings, noting variables and then trying to make sense of it all.

Deadenders



War in Middle Earth

The world of Tolkien's Middle Earth once again comes to the computer screen only this time as a wargame. In it, you control the forces of the Fellowship of the Ring in a desperate fight against the evil forces of Sauron. Your objective isn't military victory, it's far more important than that, as you must clear a path through the evil forces for a hobbit to take a Ring to the cracks of Doom.

At your disposal you have the dwarves of the Iron Hills, princely elves, and men of good heart as well as the Fellowship party of Frodo, Gandalf and Aragorn. Frodo is, of course, the ringbearer and can wear it at will to escape the battlefield and the area. However, it has its dangers: as wearing it corrupts the soul and takes you into the realm of the Nazgûl who will attack you if they are near.

The main display shows the map of Middle Earth and a finger pointer with which you can zoom in on any area and view the forces in it and issue orders for them to move into battle. Eventually, forces will clash with the undisplayed

Deadendiner is an adventure based on everyone's favourite soap opera - *Neighbours*. Nah, seriously gaw, it all takes place in Herbert Square, home to the Squibs, Pringles and other unsavoury characters.

The story starts after Bill has lost his Willie. Yes, in case you haven't guessed, it's a spoof full of caricatures and one or two slightly naughty jokes.

You play the part of PC Donald Dance and your mission, should you choose to accept it, is to find out who did the dirty on Willie. The taley tells you that poor old Willie turns up squashed, and with two holes in his neck. Leave it to me, mate, the missus'll never believe this/ta. It also goes on to say that you can talk to the character that you meet. "Ask Brady about Colin" is cited as an example. I tried it, and was met with a "Ere speak, but it doesn't further your inquiries." None of the Deadendiner wanted to talk to me. Was it something I said?

The jokes are obviously geared towards *Neighbours* fans, but I can quite believe that non-followeres of the series would get at least a smile from the game. Most of the humour, for me at least, came whenever I examined a character. When I examined Scary, I was told "Purple eyes and bright green hair, she resembles a technician's dream." Great stuff. All the descriptions are along these lines, and most are less than charitable.

When it comes to gameplay, I have a few reservations. The game doesn't accept some of the normal abbreviations. I had to type in "inventory" as opposed to just "I". Not all the cuts were listed, even the ones that should have been obvious, and mapping left something to be desired. When standing on the landing of the Old Queen, I was faced with the kitchen, bedroom and living room, yet no directions were given. It was a case of "enter kitchen" etc, and no matter which of them I went into, the cut was always in the south. This system wasn't used consistently, though. From Herbert Square I could see the pub, the cafe and the corner shop. This time I had to input the word

N, S, E and W, and hope I went to the building I wanted. Some things should be taken for granted in an adventure and sticking to one way of inputting directions is on this agenda.

During the course of the game, I came across two bags, one funny and one annoying. I'd got a bag from Aylee and, whenever I tried to examine it, the game crashed. However, I was allowed to "search bag" with no ill effects. Later, I discovered that I could "examine bag" but only if I wasn't carrying it. I can just imagine how frustrating it would be for someone to find that out after going for a long time with no success! The other bag was a silly one. I'd been searching through a dustbin, as one does, and my investigations took me to the laundrette where I was reminded how disgusting my uniform was. No problem. I took it off, washed it in a machine, got it out again, then found out I had two uniforms, one wet and one dirty.

In keeping with the spirit of the game, there's no ordinary reply when you tell the computer to get something. Instead you get a chirpy "Right, me old china." Should any copies of this game reach America, it will only reinforce their beliefs that we all talk like Dick Van Dyke.

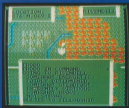
The authors have got a nice line in schoolboy humour, which I greatly appreciate. What a shame about the programming.

If you can ignore the giggles that I picked on, get down the bag an' read an' give your dach to the game in the shape. But if you're the sort who likes to be able to use single key inputs, logical geography and no bugs, knock it on the head.

One last question. Where was Kylie?

Touchline:

Title: *Deadendiner*. **Supplier:** Top Ten (Pty) Ltd, 12 Chisham Enterprise Centre, Junction Road, Phoenix, Berks RG7 4AA. Tel: 0734 267 696. **Machine:** C64.



memory and the screen will snap to the battle screen showing all the combatants, who will start the fight without your help.

You can effect the battle by using a cursor to move men into the fray or you can control one character directly. This may sound like a good idea but it doesn't quite work, as to select a character you have to position the cursor over his fort and the battles just take too long. This can be annoying, particularly if the battle was only a decoy action while the party move through the area.

Unfortunately, this spoils an otherwise interesting game which could have been so good.

Touchline:

Title: *War in Middle Earth*. **Supplier:** Melbourne House, 2-4 Grosvenor Road, Portobello Rd, London W11 2JN. Tel: 01-737 8878. **Machine:** C64/128 Price £14.99 (Disk) £9.99 (Cass).

Program Analysis

Programming can be made simple with these three C64 performance analysers



COMMODORE 64 PERFORMANCE ANALYSER

Basic is a programming language which makes it very easy for programmers to create complex programs with a minimum of effort. We pay a price for this programming ease, and that price is often poor performance; that is, our Basic program runs slowly. Another problem which confronts the Basic programmer is what to do when a program runs without failing, but doesn't give you the results you expect. How do you find out what your program is doing without adding PRINT statements to your program to trace execution or interrupt execution at strategic points?

The Performance Analyser helps to overcome both problems. Not only does it trace the logic flow in a Basic program, it also determines how long each Basic line took to execute. Thus the Performance Analyser is a generalised performance analysis tool for the Commodore 64.

Performance Analyser TRACE Facility

Most commercial tracers usually amount to a window displaying line or six line numbers on the screen as your Basic program runs. The line numbers scroll in the window as each line is executed, and the window may or may not interfere with your program output. You normally cannot trace a Basic program which uses hi-res graphics, and you certainly cannot go back and check the line number sequence previously displayed. Although you can usually close the trace display down (by the space bar for example), you have very little chance of writing down the line numbers on paper for a more detailed analysis.

The Performance Analyser overcomes all of these problems. It allows you to trace any Basic program which uses normal screen graphics, hi-res screens, sprites or sound and does not interfere with the operation of the program. The Analyser will not slow

your program down, and allows you to give the trace display at your leisure. You may scroll backwards or forwards through the line numbers for as long as you wish.

Performance Analysis

The Analyser also provides you with a tool to determine how efficient your Basic program is. When it displays the line number it also displays the time it took to execute the line. As you scroll through the line numbers you can tell at a glance which line numbers are slowing down execution and which line numbers are executed most often. Basic programs are the same as any other program - they follow the 80/20 rule. That is, 80 per cent of the work is usually done by 20 per cent of the program. The Performance Analyser is the tool you need to tell you which 20 per cent of your program is doing 80 per cent of the work, and how long it is taking to do it. You can then concentrate on making that part of your program more efficient.

Analysing a Basic program

The Analyser is written entirely in Machine Language, and is designed to cause as little interference as possible with the trace program. The Analyser is normally loaded at 38912, and all Analyser variables and constants are contained in the 2K from 38192 to 40559. Your Basic program thus has the RAM between 3048 and 38911, any low storage locations it requires and the free RAM at 49152. Should you require the RAM at 38912, then set the top of Basic pointer (35,56) to the last RAM location available to Basic, and the Analyser will use 2K of RAM before this address. For example, if the top of the basic pointer is set to 32768, then the Analyser will load itself at 80328.

Type in the Analyser loader program and save it as ANALYSER1. Make sure you verify that what you saved is correct. To use the Analyser, simply issue a load "ANALYSER1" after setting the top of BASIC pointer if necessary. ANALYSER1 will set the required Basic pointer, POKE the Analyser Machine Language logic into the correct RAM, relocate all required ML addresses and print messages to indicate how to start and stop the Analyser and display the trace data. The following messages are displayed on the screen by ANALYSER1 during execution:

```
LOADING THE ANALYSER AT
38912
LOAD OK
RELOCATION OK
1. START ANALYSER = SYS 38912
2. STOP ANALYSER = SYS 38913
3. DISPLAY DATA = SYS 38918
```

If the load fails, or the relocation of addresses fails, a message is issued and ANALYSER1 stops.

Obviously to start the Performance Analyser you SYS to 38912 or to the address displayed by ANALYSER1. You can do this from a program or from direct mode. The message TRACE STARTED is displayed by the Analyser, unless you start it from a program. The message is not issued then to ensure that the Analyser does not interfere with program messages or displays.

After the Analyser has been loaded, you then LOAD the Basic

program or programs you wish to analyse. The Analyser monitors execution of your program(s), and saves trace data in the trace data buffer for later display. If you only want to trace part of a Basic program, you would do the following:

```
1000 REM START THE
ANALYSER
1001 SYS 38912
1020 FORTH = JEEPSTER.1
1030 X = A/COS(TH)
1040 Y = B/COS(TH)/C
1050 NEXT
1060 REM STOP THE ANALYSER
1070 SYS 38913
1080 REM DISPLAY TRACE
DATA
1090 SYS 38918
1100 END
```

After your Basic program has finished, or you stop it executing, you can stop the Analyser if you want to. However, you don't stop it to display the trace data. You may leave it active to trace another program if you want to.

Obviously, to stop the Performance Analyser you SYS to 38913 or to the address displayed by ANALYSER1. You can do this from a program or from direct mode. The message TRACE STOPPED is displayed by the Analyser, unless you stop it from a program. Again the message is not issued to ensure that the Analyser does not interfere with program messages or displays.

Finally, you may display trace data at any time by entering SYS 38918 or SYS to the address displayed by ANALYSER1, and of course you may do this in direct mode or from a program. The message NO TRACE DATA is displayed by the Analyser if there is nothing to display. Again the message is not issued if you are under program control. This is to ensure that the Analyser does not interfere with program messages or displays.

If there is data to display the Analyser presents it in full-screen mode, that is a page or full screen data consisting of line numbers and line execution times is displayed and the Analyser ML program waits for you to press one of the function keys; F1 terminates the display, F5 scrolls back to the previous page of data and F7 scrolls forward to the next page of data.

You may scroll back and forward through the trace data for as long as

you like with function keys F5 and F7. When the end of the trace data is found, the number of lines executed and the total execution time is displayed, and the Analyser ML program waits for you to press a function key. The Analyser will only recognise F1, F5 and F7 function keys. All other keys are ignored. If you scroll forward from the end of the display, you wrap around to the start of the trace data again. You can't scroll back from the top of the trace data, you may only scroll forward.

NOTE: Trace data will be displayed automatically when the trace data buffer area is full. The trace data buffer is actually the RAM under the BASIC ROM. As much trace data as possible is stored there before the execution of the Basic program is interrupted and the trace data displayed. If you want your Basic program to continue, simply press F1 and the trace display is terminated. Your program begins execution from where it was interrupted. If you want to browse the trace data, then use F5 or F7 to scroll back and forward through the data.

How the Performance Analyser Works

The Analyser works by monitoring the execution of Basic programs via the character dispatch vector in low storage. As each program byte is interpreted, the Analyser checks to determine if the current line number (57,58) has changed from the previous byte read. When the line number changes, then the Analyser stores the line number and current time in the trace data buffer under the BASIC ROM. This is done until such time as the trace data buffer is full.

When the buffer is full, the Analyser saves the first 2K of low storage (0-3047), colour RAM and various control registers in the RAM under the KERNAL ROM. The trace data is then displayed, and when the display is stopped via function key F1, the Analyser restores the first 2K of low storage, the colour RAM and the various control registers. This allows the Basic program to restart execution from the point where it was interrupted, and the program screen is restored, as well as character colours and backgrounds.

If your Basic program uses the RAM under the Basic of KERNAL ROMs, then you cannot analyse it with this utility. Note also that if your

Basic program costs the time (TB = "000000"), then the Analyzer will not fail, but the execution times displayed will be appreciable.

COMMODORE 64 PROGRAM ANALYSIS

Commodore 64 Program Analysis (CMPANAL) is a Basic program which analyzes the contents of any Basic program and displays the information on the screen or printer. CMPANAL first displays summary information which contains the program name, the size of the program in bytes, the number of lines in the program, the total number of commands (e. PRINT, GOTO, IF, etc.) and the number of variables.

Once the summary data has been viewed, a detailed list of the commands used in the Basic program and the number of times each command is used is displayed. When you have finished viewing the command data, a detailed list of the variables and the use of each variable is displayed, and when you have finished viewing the variable data you may end the display, ask for the information to be redown or send the data to your printer.

Using CMPANAL

CMPANAL allows you to analyze your Basic programs. It does this by running in the 4K of free RAM at 49152 to 52387, and loading the Basic programs it analyzes at 2049. By not using the RAM between 2048 and 49943, CMPANAL is capable of analyzing the largest Basic program. However, with only 4K of RAM to run in, CMPANAL will run slowly analyzing large Basic programs because many garbage collections will be done to ensure that there is sufficient space for CMPANAL to operate correctly. Also, only 50 variables can be displayed because of space constraints.

Obviously if CMPANAL is to run in the RAM at 49152 then some changes need to be made to Basic pointers in low storage. The start of Basic and end of Basic addresses need to be changed as well as the start of variables etc. These changes are handled by the CMPANAL loader program. LOADER is the Basic loader program which automatically loads

CMPANAL. It sets the low storage pointers, and then uses the dynamic key facility to automatically load CMPANAL.

You must create and save LOADER first on tape or disk. Next type in CMPANAL and save it directly after LOADER on tape or on the same disk as CMPANAL.

Note that if you are using disk you need to change line 10 in LOADER from LOAD "CMPANAL", 1,1 to LOAD "C:\CMPANAL", 8,1 so that CMPANAL will be loaded from disk and not tape.

Once you have saved LOADER and CMPANAL to tape or disk then simply load LOADER and RUN it. LOADER will set the various low storage pointers and then sit at the screen and keyboard buffer so that when it ends, CMPANAL is automatically loaded at 49152. When CMPANAL has been loaded it begins execution automatically, clears the screen and places the first message on the screen:

LOAD FROM DISK (Y/N)?

If you want CMPANAL to load the Basic program it analyzes from disk, then reply Y. Otherwise reply N and the program will be loaded from tape. Before replying to this message, you should have the tape or disk which contains the program to be analyzed in the cassette or disk drive.

The next message to be displayed is:

PROGRAM TO BE LOADED = ?

Your answer to this message with CMPANAL the name of the program it is to load from tape or disk to analyze.

CMPANAL then uses the KERNAL load subroutine to load the Basic program at 2049 and begins to analyze it. Since it may take some time to analyze large Basic programs, CMPANAL places the line number being analyzed in the top left-hand corner of the screen while scanning the Basic program. When analyzing is finished the summary report is displayed as follows:

```

---PROGRAM STATISTICS---
PROGRAM NAME = CMPANAL
PROGRAM SIZE = xxxxx
NO OF LINES = xxxxx
NO OF COMMANDS = xxxxx
NO OF VARIABLES = xxx

```

USE ANY KEY TO CONTINUE

You may view the summary report for as long as you wish. To move to the command report, simply use any key and the following display appears on screen:

- COMMANDS -

```

END          = 1
FOR          = 3
NEXT         = 6
DATA        = 100
INPUT       = 1
READ        = 1
GOTO        = 125
IF          = 30
GOSUB       = 17
RETURN      = 17
REM         = 8
POKE        = 5
PRINT       = 28
THEN        = 25
+           = 45
-           = 15
*           = 31
/           = 19
AND         = 1
"           = 45
MID$       = 3

```

USE ANY KEY TO CONTINUE

If all commands used in the program can be displayed on one screen, then when you press any key you will move to the VARIABLE display. If more commands are used than can be displayed on one screen, then the next screen of data will contain command data. When the list of the command data has been displayed and the USE ANY KEY message is displayed, when you press any key the list of variables appears on the screen. Note that +, *, /, =, <, and > are considered comments when used in statements such as A+B<C/D/E/IF X GOTO 1000.

When the commands are finished, the list of variables is displayed as shown:

```

- VARIABLES -
I           = 2
X           = 4
RRS        = 5
Z           = 3
Z$         = 6

```

USE ANY KEY TO CONTINUE

When the list of the variables has been listed, CMPANAL displays the following message:

R = RE-DISPLAY, X = END, P = PRINTER

If you press the R key, then all information beginning with the summary display is redisplayed. If you press the X key then program execution is terminated and the final time message is displayed:

TIME TAKEN = XXXX.XX

This is the time in seconds it has taken C64PANAL to analyse your program. You may then use C64PANAL to analyse another Basic program. Press P and the information is sent to the printer.

Applying C64PANAL

C64PANAL has many uses. You can find the size of your Basic program, the number of variables you use and the number of lines in your program. The number of lines is important, because each line in a Basic program carries an overhead of 4 bytes (2 bytes for a link address and 2 bytes for the line number). A 300 line program uses 2,000 bytes of storage for link addresses and line numbers. If you have an excessive number of lines, you can conserve space by reducing the number of lines (also known as crunched your program). You reduce the number of lines by placing multiple commands on the same line separated by colons, removing blanks and removing REM commands.

Processing new lines also carries with it a performance penalty. The more lines in a Basic program, the longer it normally takes to run. By reducing the number of lines, you normally reduce program execution time. C64PANAL will tell you how successful you have been at reducing the number of lines in your program. It will give the size of your program and the number of lines before crunched, and then after you have made your changes you can run it again and get the new figures.

The detailed list of commands (ie PRINTs, GOTOs, IFs etc.) can also be used to reduce program size and increase performance. For instance, if you find that you have a very large number of IF commands, then you may be able to reduce them by using the ON command. For example if you have:

```
IF CC = 1 GOTO 1000
```

```
IF CC = 2 GOTO 1100
IF CC = 3 GOTO 1200
IF CC = 4 GOTO 2000
IF CC = 5 GOTO 2100
IF CC = 6 GOTO 2200
then you could replace the IF com-
```

```
ON CC GOTO 1000,1100,1200,2000,2100,2200
```

It is also interesting to see the pattern of commands in various programs and which commands are used most frequently. In string operations the LEFT\$, RIGHT\$, MID\$ etc. will figure prominently. However, the most common commands used are the IF, GOTO, FOR and NEXT and PRINT.

The list of variables is a powerful tool to help in the creation of your Basic program. Basic maintains a list of variables, and the closer a variable is to the start of that list, the less time that is needed to find the variable when it is referenced in a statement. For example, every time IF X = 3.7 GOTO 100 is executed, the X variable must be found in Basic's list of variables to check if it is 3.7 or not. Thus the closer X is to the top of the list, the faster it is found. The order of variables makes a significant difference to the execution time of your program if you have a large number of them. C64PANAL helps by giving you a guide as to which variables ought to be defined first so that they appear at the top of Basic's list of variables. You can ensure the order of variables by defining them in the following order:

```
X = 0:A = 0:Y = 0:PC = 0:TK% = 0
etc.
```

X will come first, A second, Y third in the list and so on.

If you have Basic programs where execution time is crucial (for eg. games programs) then C64PANAL will be an important tool to help you analyse those programs and make them faster.

COMMODORE 64 SWITCH

Commodore 64 SWITCH is a short Machine Language (ML) program which resides in RAM just before the BASIC ROM. It occupies storage locations 40704 to 40696. C64 SWITCH allows you to partition your C64 into two logical machines. You switch between the two partitions or regions with a single key

stroke. With this utility, you can load two Basic programs at once and compare them or work on them. However, you cannot have both programs running simultaneously.

Using C64 SWITCH

C64 SWITCH allows you to set variable region sizes. The regions are designated zero (0) and one (1) and region 0 will extend from location 3049 to the limit you set, while region 1 extends from the end of region 0 to 40704.

To use the switching function, simply load SWITCH, which is a Basic loader program. When you run it, SWITCH will load the ML routine at 40704 and display the message:

```
ENTER REGION # ENDING
ADDR = ->
```

You enter the ending address for region 0 (and then region 1 starting address) and the final message are displayed:

```
REGIONS 0 and 1 INITIALISED
REGION ACTIVATED = 0
```

To switch between the two regions use the F1/F2 keys. F1 will activate region 0 while F2 will activate region 1. The active region is displayed in the upper right-hand corner of the screen in reverse video. To deactivate the SWITCH, simply hit RUN STOP/RESTORE or turn the C64 off and on.

Applying SWITCH

C64 SWITCH has three main uses. You can load two Basic programs at once, and work on them or compare them. You can use region 1 as a data region which is accessed by a program in region 0 (SWITCH was originally written for this purpose). Finally, you can use SWITCH as a means of merging two programs. If you want to add code to a program in region 0 from a program in region 1, simply LIST the statements in region 1 on the screen, press F1 to activate region 0 and then move the cursor over the lines you want added and press RETURN. Each line will be entered into the program in region 0.



Help!

Enhance your help function with this handy utility

By Mark Everingham

In the old Commodore advertisement (you know, the four page epic which managed to link Charles Babbage, an elephant, and a teddy-bear called RJ to buying a Commodore computer for Christmas), special emphasis was laid on the 'HELP' function of the C16 and Plus4 computers. Commodore claimed it helps you to debug your programs, yet I have owned a Plus4 for several months now, and can honestly say I have never used the HELP function except for the novelty. The BBC Micro has a command *HELP which lists the syntax of a given command on a sideways ROM, and I decided to implement such a function on the Plus4. I decided on three features it should have:

1. It should be compatible with the Commodore C16.
2. It should not take any memory from the programmer.
3. It should not interfere with the normal HELP function.

A tall order? Well, I decided that to allow a reasonable amount of help on the C16, the program must use the Disk Drive. That way, I could put it in the cassette buffer so as not to use up any memory; and a CHERGET wedge like the DOS SUPPORT program seemed appropriate to allow for the normal HELP function. The result is a 143 Byte piece of machine code using standard PRG files on a Commodore Disk Drive such as the DSII.

The Programs

Listing 1 is a short and sweet Screen Editor I wrote to produce HELP screens and store them on Disk.

Listing 2 is the Basic Loader program for the HELP command. It Pokes all the code into memory and redefines the function keys.

Listing 1 - The Screen Editor

When you've entered and debugged the program, save it onto a disk using `DSAVE 'HELP EDITOR'` and `RUN` the program again. You should be presented with a white screen showing the usual flashing cursor in the top-left corner. At this stage, the editor acts just as if you were editing a document - type text, in normal or reverse, graphics symbols, anything you want, even use the ESC functions to format your screen.

Pressing `RETURN` from the first level puts you into the command mode. A bar will appear at the bottom of the screen with three options. Press the relevant function key to select each one.

`LOAD` prompts for a filename and attempts to load this HELP screen from the disk.

`SAVE` prompts for a filename and saves the current screen to disk under that name.

`CONT` puts you back into level - the edit mode.

When you've created a help screen, save it onto Disk under a suitable and memorable name, and exit the program. The HELP command now has something to work with. If you `DIRECTORY` the disk, you will see a file filename.H. The 'H' designates a HELP file.

Listing 2 - The Basic Loader

Now that you have some data on disk type in Listing 2, the Basic Loader, and save it on disk using `DSAVE 'HELP PROGRAM'`. When you have run it, try pressing function key 1. You should see the message `SYS 913:REM ON?`. `SYS 913` turns the HELP command on, press function key 2 or type `SYS 996` to turn it off.

With the HELP Command on, try typing in `HELP (RETURN)`. This



simply does the normal HELP function - if your program has no errors in it, nothing should happen. Now, try typing `HELP 'Filename'`, where `Filename` is the name you stored your Help screen under. If all is well, your screen should load and the `READY` prompt will appear at the bottom of the screen. If this does not happen, something is wrong - if you get an error message `'STRING TOO LONG ERROR'` this means that you have tried to type in a name more than 14 characters long. If you get the error message `'DIRECT MODE ONLY ERROR'` then you have tried to use the new HELP command from within a program. If neither of these, see PRINT D33 to find the error. The Syntax, and errors returned by the HELP command are shown below.

Syntax of the HELP Command

`HELP (RETURN)` - Normal HELP function

`HELP (Filename)` - Loads help file called 'Filename' from disk
`STS 900` - Turns HELP Command On
`STS 990` - Turns HELP Command Off

Errors returned by the HELP Command

All DOS Errors - a fault concerning the disk
`'STRING TOO LONG` - You have typed a Filename longer than 14 Characters
`'DIRECT MODE ONLY` - You have tried to use HELP in a program

Information on the HELP Program

The HELP Command resides by default in the first 142 bytes of the cassette buffer (BIO-32BF). Note that when it is installed in this area of memory, the program will be erased by pressing 'RESET'. I placed it here to avoid clashes between the Plus/4

and the C16. However it may be relocated by changing the address A in line 9 of the BASIC Loader Program.

Practical use of the HELP Command

When creating HELP files, try to give them memorable and unambiguous names. It is also a good idea to make them short, though the " and ' designations may be used in filenames. For instance, if you are going to replace the manual with a HELP disk, divide the files into commands, each showing the syntax and a few examples of the command's use. E.g. Typing `HELP 'CIRCLE'` might bring up all the different ways of using the command `CIRCLE`. The thing of utmost importance is to "use your common sense" in a system which is very powerful when used properly, but could end up not being helpful at all if help files are designed without thinking. Anyway, I hope it will be very useful to you!

GAMES UPDATE

Super Cycle



The budget becomes continuous with this classic bike racing game that topped the charts in its full-priced format. Myriad Controls wait to test run and machine

in a battle against other riders and the clock, as you have to cross the line before the seconds run out to qualify for the next round.

The early tracks are easy enough, and although the other riders can get in your way and even cause a crash, you'll easily catch up the time. Later on things get tricky, with the addition of slippery road surfaces to avoid you sliding, treacherous icy conditions, obstacles such as hurricanes and cones to send you crashing, and a tighter time limit where every second will count.

To add variety to the races, the swirling landscape changes from day to night, country to city, and mountains to desert, the most impressive being the race near Cape Canaveral.

Avoids action at its best.

Touchline:

Title: Super Cycle. Supplier: Atari (US Gold's, Data 2/3, Wolford Way, Hatfield, Birmingham, B6 7AJ. Tel: 021-336 336336). Price: £2.95.

The Deep

If you tried to imagine what a game called *The Deep* would be like, you'd probably think of barrel-rolls, sharks, octopuses or perhaps submarines.

You certainly wouldn't think that you'd spend most of the game on the

sea's surface! Unfortunately, this is exactly what you do in this rather disappointing game.

It reminds me of a very old arcade game which I think was called *Submarine Attack*, in which you controlled a ship armed with depth charges that had to survive an assault of submarines that released mines and fired torpedoes at you.

In *The Deep*, your task is to destroy subs, after which small flags are occasionally released and float to the surface, staying there for three seconds. If you manage to avoid the floating mines and reach them in time, a helicopter will fly overhead and drop a package that you must catch. These packages increase your chances in the game by supplying hydrofoil speed to your ship so you can skim across the surface, smart bombs to wipe out all subs, sonar missiles that can be aimed and destroy everything in their path, or submersible pods that can collect the tokens that



lie on the sea bed to clear a screen.

There are three transition games in which you must destroy a giant ship, a massive submarine and protect a fleet of escaping hostages by shooting down the missiles that are being fired at them, but then it's back to the main game again.

Touchline:

Title: The Deep. Supplier: US Gold, Data 2/3, Hatfield Way, Hatfield, Birmingham, B6 7AJ. Tel: 021-336 336336. Price: £2.95.



GRAND PRIX CIRCUIT

Despite the split between Electronic Arts and Accolade, the string of Accolade imports continues from the EA stable. This latest one attempts to recreate the world of Formula 1 racing, and offers you the chance to drive for either the Ferrari, Williams, or McLaren teams in a world championship against nine other drivers and over eight Grand Prix races.

Selecting the team you will drive for also selects the type of car you will drive - for example, the Ferrari is slower than the others, but is not as likely to spin and so is a good car to start with. When you think you can control the car, you may want to swap to a Williams or the fastest, the McLaren. A more detailed appraisal of each car is presented on-screen in a display that shows the power curves or horse-power and torque, engine size, gear box, chassis and weight, so that people who think these figures are important can use them to decide which car to use in the championship.

Perhaps of more importance is the game level you choose to play at, as this can determine whether it will be a quiet, sporting race without any mechanical problems, or a bitter fight where the best car and driver will win. The problem with the car is keeping it going at a speed fast enough to maintain your race position, but slow enough to stop it from blowing up.

Tyres are also a major headache, particularly in the longer Grand Prix, where cornering too quickly and spinning can cause serious wear. They can be changed in the Pit, but this can cost you valuable seconds if your Pit team isn't quick enough.

The first Grand Prix of the season is at Rio de Janeiro in Brazil, but before you can think about the nine championship points for winning it, you have to drive round

the circuit in a qualifying lap. This not only gives you a preview of the track, but the time you take decides your position on the starting grid.

The race screen display shows your view of the track, and your controls, which include a tachometer to watch for engine revs, a damage indicator that plots the condition of your car, especially braking and handling, a speedometer, rear view mirror to check that no-one's in your sight-rear, and a map box that includes a line drawing of the course and a flashing dot to plot your position.

Steering can be a little tricky at first, as moving the joystick turns the wheel, and you must remember to turn it back again as it doesn't auto-centre, unlike most driving games. However, once you've mastered the basics you have to contend with the competition, particularly at the higher game levels when any collision will put you out of the race.

The corners are the best place to overtake, where the driver with the straggled nerve will take the lead - it's all down to when you apply the brakes. If you brake after your opponent, you'll go round the bend in the lead, but if you have it too late you'll spin off.

While Grand Prix Circuit is a good simulation of a formula 1 championship, an eight race season will probably prove too much for all but dedicated formula 1 fans. There is an option to race in a single Grand Prix, but there are better racing games for the casual driver. This one's for those who live on carbon monoxide car fumes.

Teacher:

Title: Grand Prix Racing *Supplier:* Accolade (Electronic Arts) *Langley Business Centre, 12-49 Station Rd, Langley, Bucks, SL3 8TN. Tel: (0753) 4041.*



Infiltrator



Here's a second chance to fill the empty boots of the one and only Johnny "Jamba Baby" McGibbets, as he flies off in his Whizzbang Enterprises Gismo chopper to save the world at least three times. The former full-priced game has been re-released via the Kixx label, and it's excellent value for money.

His opponent in all this is the aptly described but unnamed Mad Leader, who is threatening all sorts of

disgusting things if he isn't stopped, and Jamba Baby's the man for the job.

Each of the three missions follows a similar pattern: a three stage game in which you must fly through enemy streptococci to find the Mad Leader's camp, then (you float) infiltrate the heavily armed camp and search the buildings for secret plans, weapons and so on.

In fact, *Infiltrator* is three games in one, starting with a combat flight simulator in which you must combine stealth, to confuse the enemy, with fighting skills to shoot them down. In the camp, you are armed with laser papers and sleeping gas to get into the camp without raising the alarm. Inside the buildings, things get tougher as you must avoid the guards and search (by possible Monty-style) every object in every room. If the guards get too noisy, shouting that your papers may betray them, if not, use some sleeping gas and then get out before the game round and raise the alarm.

Infiltrator was a smash hit in the States, but is still underappreciated over here. At £2.99, it's a bargain.

Footballer

Title: *Footballer*. **Supplier:** Alex (US Gold). **Disk:** 2/3. **Walford Way, Melford, Birmingham, B6 7AE. Tel:** 021-335 3354. **Price:** £2.99.



This is the re-release of the conversion of the original arcade game that began the chase for car shoot-outs that has resulted in a string of games including *Road Blasters* and *LED Skies*. It all started in 1983 with *Spy Hunter*.

Naturally, it looks a little dated with its top-down view of a scrolling road network that you must patrol, but the gameplay is just as addictive, and has certainly stood the test of time.

The action begins as the weapons van pulls up at the side of the road and the spy car rolls out into your joystick control and begins with a machine gun as its only weapon. Your mission is to stay alive as long as possible (oddly enough) and clear the road of the villains - Road Lords, Barrel Dumpers, Enduroers, Switch Blades, and Mad Bombers (helicopters).

These villains attack you with appropriate weapons, which you must answer as best you can, while staying on course on a road that turns and splits to confuse you. At some stages the road disappears altogether, but hardly your *Spy*, as there becomes a *Spy* boat and takes the battle on to the waters.

Destroying enemies not only racks up the points, but also earns you reinforcements with the weapons van and an upgrade for your weapon system, which includes rockets to take out bombing helicopters, and oil slicks and smoke screens to deter enemies that get too close.

Even after six years, *Spy Hunter* is still a great shoot-out.

Footballer

Supplier: Kixx (US Gold). **Disk:** 2/3. **Walford Way, Birmingham, B6 7AE. Tel:** 021-335 3354. **Price:** £2.99.



Denaris is a planet with a problem. For years its scientists have been developing super-advanced machines, and so it was almost inevitable that one day the machines would grow so advanced that they wouldn't need the men any more.

By the time the Denarians realised what had happened, it was too late to launch a direct assault. They tried anyway, but it just made the machines more powerful. Their only chance to break the tyranny and escape from their underground prison is you, living a small fighter.

The fighter is, of course, highly manoeuvrable, and can be impressed by collecting items and debris from enemy ships. Before you start thinking that you've heard it all before, and that this is just another *Arkanoid* clone, I'm delighted to tell you that it isn't. For one thing, two of you can play together.

The fighter is a DS-H72 Eagle Fighter that begins the game unprepared for the onslaught ahead. You'll face a barrage of meteorites, swarms of mechanised fighters and have to brave the firepower of land-based gun emplacements. The other player controls a satellite that's activated by player one collecting a crystal early in the game. From then on, it can either move independently or dock with the fighter (which is what happens in the one player game).

As with *Arkanoid*, you can pick up objects to improve your ship, but these are harder to come by. They have specific effects on your ship, and come in the shape of crystals, extra weapons and bombs. Crystals are split into two groups, with the Derga crystal activating and improving

the firepower of the satellite, and the Zerkis stars increasing and decreasing the speed of the alien ships.

There are five types of balls that add a cumulative effect on your ship – they include a red ball to increase shooting power, green to add to the number of missiles, blue to activate a temporary shield, grey to add 1000 bonus points and yellow, which acts as a smart bomb destroying everything currently on the screen.

Extra weapon symbols are less common, and include systems known as the Scatter Shot, Lightning Bolt Shot and Power Shot. You can also collect protection sandbars that fly above and below your fighter and deflect incoming missiles. You will need all this firepower and protection if you are to survive the level, as well as the inevitable missile machine that guards the way to the next screen.

Perhaps the most important thing to learn is control over the fire button, as this is not a game where an unrelenting joystick is the way to get a high score. Instead you must choose the right moment to push the fire button for a short burst of energy, or press and hold the button to build up the power that can take out a medium-sized asteroid or dent a super machine.

Obviously, *Denaris* is inspired by games like *Arkanoid*, but includes enough subtle differences to make it worth a second look.

Troubleshooter

Title: *Denaris*. **Supplier:** Rainbow Arts (US Gold), Unit 213 Watford Way, Watford, Birmingham, BB 74LE. Tel: 01923 339 3399. **Price:** £4.99.

3D Pool



Pool and snooker games are very popular, but they've always had the problem of being played entirely from a top-down perspective, making it difficult to shoot accurately. Now, thanks to Firebird and the skills of pool champ 'Malrose' Joe Barakara, you can get into the action in one of the first true 3D pool games.

It's quite a remarkable piece of programming - you can walk around the table, pull out for an overhead view and then get down close to aim your shot. There are no aiming lines or cursors, it's up to you to get down close and judge the angles.

You'll need to practise a little on the controls before you can expect to clear the table in a single break. Left and right controls move you around the table, while up and down zoom in and out. Pressing the fire button up and down increases and decreases the power of the shot, moving it left and right controls the horizontal position where the cue will strike the cue ball. This decides the "side" as the ball, and whether it will curve left or right after it's hit another ball as a cushion.

Top or spin is added by hitting the cue ball above or below its centre, and this, curiously enough, is controlled by the same controls that move you in and out. Therefore, to spin the ball back, you have to look down on the table from overhead.

Eventually, you'll sort that cue out and be ready to take part in the tournament against seven other players for the right to play Malrose Joe himself. When you realise that Joe is pool's equivalent of Steve Davis, you'll appreciate why you have to qualify first.

This is also the first C64 pool or snooker game which includes an actual tournament where each round is played over more than one frame. For example, you begin your challenge in the quarter final in a best-of-three match, then it's best-of-five in the semi, and seven in the final.

The computer opponents provide a reasonable game, based on the fact that they know what they're doing, whereas you're learning. As you get better, you'll get closer to the challenge match with Malrose Joe. If that's too much for you, or you find it too easy, then you can opt to play against another human, or try one of the 12 trick shots with which you can amuse your friends.

The game plays the popular pub version of pool, in which you must pot all seven of your coloured balls before sinking the black to win the match. However, any foul shots (and the computer opponents occasionally play what I would describe as deliberate misos) are punished by awarding the other player an extra shot and a free ball.

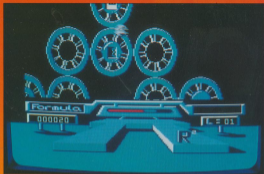
This game is definitely a cut above the usual 2D style of pool games, but it does have some rough edges. Although the balls have shadows and reflections, they slide rather than roll across the table, but that's probably more to do with the technology of the hardware than the programming. The oddity thing is the lack of cushions. Visually, they're there, but the balls seem to drift through them and hang off the edge of the table. However, I doubt there'll be too many complaints about these minor graphical points.

As a first 3D pool game it is exceptional and far from the ridiculous seven cushion shots that plague many 2D versions.

Firebird:

Title: 3D Pool. Supplier: Firebird, 84-76, New Oxford St., London, WC1A 1PS. Tel: 01-367 7847. Price: £9.99 (inc.) £12.99 (retail).





Bargain Bucket!

Six new budget games of varying quality are given the once-over by Gordon Hamlett

Four titles this month, offering six different games, so everybody should be able to find something to suit both pocket and taste.

Missing Omega from Big Hyle sees you trying to explore and deactivate an alien object - Omega. The problem is, you only have one hour of real time to accomplish this. There are four different situations that need to be shut down in this period in order to ensure the safety of life, the universe and everything.

You start off by designing your own robots, each made up of four different components - base, weapon, sensor and pointer. You must then explore Omega, controlling your robot either manually, on automatic or by programming it.

This is an intriguing game, but it's let down somewhat by an inadequate set of instructions so that even after prolonged playing, I still had little idea of what exactly I was trying to accomplish.



Three games for less than a pound each is the boast of *ZZZ* from Silverbird. Pick of these titles is no doubtably *Road 'n' Wheels*, your chance to try out such moves as flying, body pressing and aeroplane spins on the likes of *Vicious Vision* and *Georgous Greg*. Bouncing off the ropes to build up speed, you should attempt to soften your opponent up a bit before attempting to pin him to the canvas.



The other two games are somewhat less noteworthy. *BMX Rally* has you trying to qualify for the next race by finishing in the top three in your current one. It is not sufficient just to complete the course, you have to perform stunts and tricks as well. Collect cans of pop to boost your energy and wheels to repair the damage caused by collisions with the other riders.

The final game is *Ninja Werner*, which is a truly dreadful martial arts game, but one so bad that it's almost worth having for that fact alone. Dodge arrows, karate chop legs, fend off death stars with your sword and finally shoot down samurais with your blow-pipe, if you can master enough moves.

J.R.Squared, also from Bag Byte, is a strange game. You must chase round eggs trying to pick up the different elements of assorted mathematical formulae. Pursuing you round the perimeter of the eggs are the chasers. Contact with these decreases your IQ, although you can restore this by collecting bonks. Other objects may be beneficial or hazardous to your journey. I didn't particularly enjoy this game first time round, and I'm afraid that time has not mollified my opinion.

The final game this month, and by far the best of the bunch, is *Don Goro II* on Mastertronic's Riscobar label. Perhaps the most notable aspect of this sequel is that it is actually different from the original - something most unusual in this industry. The evil Melkon has developed a race of Supertrons, and it's up to Goro to penetrate the four levels of the Melkon's ship and sabotage the control bases.

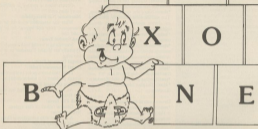
You can also choose to play the Melkon, attempting to mutate the Supertrons into outer space. In either case, you only have a limited amount of time to accomplish your task before moving onto the next level. This is a good-looking game that plays exceptionally well, and if you don't already have it in your collection, I suggest that you go out and pick up a copy straight away.



Bargain Bucket!

A powerful 6502 assembler for both tape and disk users.

By Steve Currie



The ASM Assembler

The ASM assembler is a dual mode system whereby 6502 assembly language programs may be compiled from either one or more disk files and/or memory. It provides a set of commands to control its various functions, and also allows output to a Commodore printer.

The system has two modes of operation: disk mode and memory mode. In disk mode, the source code file is read from the source disk drive, and its compiled output sent to another file on the destination disk drive. In memory mode, the source code is read from memory (where it is edited in either mode) and output to memory. Code relocation facilities exist to allow a program to be assembled to run at one address but placed at another.

The ASM system has two error modes, fatal and nonfatal. In fatal mode, an error will cause the assembler to halt, whereas in nonfatal mode, the assembler will process the whole file, listing errors as it goes. Some facilities also exist for control of disk drives and error reporting from these devices. A printer redirection facility allows all output which is normally sent to the screen to be sent to the printer at device number 4 or 5. Source and destination drive numbers may be set before assembly commences. Whilst using the editor, any Basic direct mode command may be issued.

Whilst ASM was designed primarily for use with disk drives, the fact that it supports memory assembly allows tape users to make use of it. The memory mode was originally

designed to allow short routines to be tested without having to resort to disk usage. Even in memory mode, you may still make use of files from disk as the file-disk/file-include facility still works.

Getting it all in

Listed here as a Basic loader program, ASM represents a considerable typing task. You may type in the program directly as it stands, but don't run it until you have read the next bit! Before running, (assuming you have saved a copy to tape/disk first) execute these direct BASIC commands first. These will set the memory configuration correctly.

POKE 43, 01 :POKE 44, 15 :POKE
5960,0 :NEW

Now reload the Basic loader and run it. ASM will be POKE'd into memory at the correct address and saved to whichever device you set the device no. to (1 or 4/2/16/1) see the listings. Reset the machine and load and run ASM. You should get a sign-on message and a flashing cursor.

When ASM is loaded and run, it installs a small wedge into the BASIC system. This has two important effects:

- (1) Edited program lines are no longer tokenized by Basic. This means that you cannot edit a Basic program. This is similar to the EDIT program in my 'Constructing a Compiler' series in a previous issue of *Your Commodore*.
- (2) A set of additional commands are introduced via the special character *. These commands allow you to easily access the facilities provided.

The additional commands

The extra commands are as follows:

Assembly:

***assemble** Start the assembler in the mode set by the mode commands ***mem** and ***disk**. The operation is as follows. In disk mode, you are asked for a filename whose default extension is .asm. The output file will have extension .asa. In memory mode, the source code is expected from memory and output is to memory. During assembly, output of messages, listings, etc to the list device will follow the mode set by commands ***printon** and ***printoff** whilst the treatment of errors will be defined by ***fatal** and ***nonfatal**. Assembly may be halted at any time by pressing the RUN/STOP key.

Assembler control commands:

***disk**: Sets assembler disk mode. Source code is expected from disk and output is to a disk file. (see ***assemble**).
***mem**: Sets assembler memory mode. Source code is expected in memory and output is to memory.
***fatal**: If an error is encountered during assembly, the assembler will stop.
***nonfatal**: If an error is encountered during assembly, it is displayed but assembly continues with the next line of source code. Note that output is still

produced in this mode but it should be noted that the program may not run correctly.

Information and editing commands:

***restart**: Restart a source program in memory starting at 10 and going up in steps of 10.
***symbolic**: Display the symbols being from the last assembly operation.
***show**: Show the mode of operation. This will show the assembly, error and list device modes as well as device selections.
***help**: Lists all the available commands.
***reset**: Set the ASM system to default startup mode. All values are set to default status: i.e. printer off, fatal error mode, memory assembly.
***info**: Display assembly information from last assembly operation.
***stat**: Display source device directory.
***dest**: Display destination device directory.

Device control commands:

***source**: Set source device. Argument is the device number which must be in range 0 to 11.
***dest**: Set destination device. Argument is the device number which must be in range 0 to 11.
***send**: Send a command to source device. The command must be in quotes, e.g. ***send "mkdir,d"** will format a disk.
***mkdir**: Same as ***send** but for the destination device.
***err**: Display device error for source device.
***err**: Display device error for destination device.
***printer**: Enable printer output. Argument is the select code for the list device (4 or 5).
***printoff**: Disable printer output.
Note that certain functions such as listings are controlled from within a source program; e.g. **sym**, **list** (see ***directives**).

Available Operators.

The following are valid in an expression:

- +** Has value, e.g. **asa\$C000**
Askl value, must be two e.g. **lda a "d"**
- <** Less byte, e.g. **lda e < symbol**
- >** High byte, e.g. **lda e > symbol**
- +** Addition e.g. **symbol eqn number+2**
- Subtraction e.g. **symbol eqn number-5**

Errors and their meanings

The following is a list of the various error messages which may be printed during assembly:

Undefined Symbol Error:

This occurs if a symbol has been referenced but has not been defined.

Redefined Symbol Error:

Occurs when a symbol is defined more than once.

Mnemonic Not Recognized:

What the assembler took to be a mnemonic does not appear to be a valid one.

Bad Symbol Error:

Something is wrong with a symbol. Typically an invalid character or it is too long.

Illegal Operand Field:

Illegal Macroscopic Field:

These two errors point to a general syntax problem in a source code line.

Missing Operand Error:

An operand was expected but was not found.

Disk File Error:

General failure of disk system.

Syntax Error:

A problem with a directive is likely.

Illegal Quantity Error:

Some overrange condition has occurred, typically a 16-bit value in a byte mode instruction.

Illegal Addressing Mode:

An instruction was used in an incorrect way.

Not X or Y Index:

Only X and Y index registers are valid.

Symbol Table Full:

Pretty fatal one this. It indicates that the space set aside for symbols has been exceeded.

Branch Range Error:

Branch instructions are relative and may only operate within a certain range.

Linkfile name length error:

The argument to an **link** directive is too long.

Linkfile name missing:

The argument isn't there at all!

Bad directive in memory mode:

You have used some directive not valid in memory mode.

Bad directive in disk mode:

You have used some directive not valid in disk mode.

Cannot open another linkfile:

Trapping error message when you try to **link** another file whilst already linked.

No such select code for this device:

You have tried to assign a device

```

10 : Example A
20 : Memory Mode
30 :
40 :>= $0000
50 :lt
60 :
70 :index etc $10
80 :char etc index+2
90 :screen etc $0400
100 :
110 :lda #$screen
120 :ldx #$screen
130 :sta index
140 :stx index+1
150 :ldx #0
160 :ldy #0
170 :lda char
180 :fill sta (index),y
190 :dex
200 :bne fill
210 :inc index+1
220 :dex
230 :bne fill
240 :sta
250 :
ready:

```

```

10 : Example B
20 : Using .inc file
"VARASM"
30 : use in memory or
disk mode
40 :
50 :>= $0000
60 :lt
70 :
80 :inc "var.asm"
90 :
100 :lda #15
110 :ldx #4
120 :sta vic+221
130 :stx vic+220
140 :sta
150 :

```

ready:

Type This In And Save to disk

```

10 : Module "VAR.ASM"
20 : Example for Example B
30 :
40 :vic etc $0000

```

ready:

code other than 4 or 5 for a printer ("printing") or a code rather than 3, 4, 10 or 11 for a disk drive ("source," "dest"). Occurs in editor mode only.

Device Communication Failure:

Communication to a disk drive failed. May indicate wrong device number. Equivalent to Basic's "Device Not Present" error.

Things to look out for...

When ASM is installed and running, the following information is relevant. The Basic cheat code is inserted to a new routine within the ASM code to allow the inclusion of the new commands. The program loads like a Basic program into memory starting at address \$0801. When it has been run, the start of Basic is shifted up to about \$2500. You may still type any Basic direct command such as LOAD, SAVE, POKE, etc but caution is advised using POKE on addresses between \$0801 and \$2500.

In both memory and disk modes, code is edited above about \$2500. In memory mode, the symbol table begins in memory after the source program. This also applies in disk mode hence any program in memory will be preserved. This means that you should type "new" before commencing a disk mode assembly to maximise symbol space. During assembly, the BASIC ROM is switched out. The space from \$C000 to \$CFFF is left free. Symbols that occupy the space from the end of any program in memory up to \$BFFF.

ASM should operate perfectly with the Basic interpreter. The "reset" command may be used to resolve certain situations where the system is not operating correctly. However it has a limited effect, and it may become necessary to powerdown should the system still operate incorrectly.

ASM is source code compatible with my earlier PCL system assembler published in a previous Your Commodore and also my PLUSA assembler. ASM's facilities are in effect, a superset of the PCL assembler's facilities and ASM could therefore replace the PCL assembler if desired.

To help you become familiar with the system, I have included some example source files listing which may be assembled using ASM. The comment fields at the beginning indicate which mode they should be run in. Good luck!

DMA Assembler Directives

- BYT** Byte value directive. Single values or strings in single quote values.
e.g. byt \$1, \$4, \$81, 'abcd', 0
- WOR** Word directive
e.g. wor \$C000, \$E00A, \$1234
- EQZ** Zero page square. Used to assign a zero-page value to a symbol.
e.g. pointer equ \$FB
- EQA** Absolute square. Used to assign an absolute value to a symbol.
e.g. vic equ \$D060
- ORG** Set code origin. In disk mode - also sets code load address.
e.g. org \$C000
- RES** Reserve memory.
e.g. res 65 (reserve 65 bytes)
- LST** Causes assembler to list during pass 2
- SYM** Causes assembler to display symbols upon completion of the assembly.
- LNK** Chain to another file. When file has been assembled, the current file resumes assembling.
e.g. link "symbols.asm"
- REL** Relocation offset. The code origin is set by the org directive. This directive allows you to assemble code to run at one address (org) while being sent to another memory area (rel).
e.g. org \$0000
rel \$C000

Disk Edit

Delve further into your disks with the help of this article.

By Fergal Moore

Disk editing is what separates a casual disk user from a professional. Once you can edit disks, a whole world of seemingly impossible tasks becomes possible. Files can be locked, unscratched, closed, relocated, and renamed when you have the commands and the know-how.

Firstly, a word of warning: don't edit a disk with important programs on it, unless you know what you're doing. Use unwanted disks for practice, and take backups of valuable disks. A good Disk Editor will make things a lot easier: you have no need for complex commands. There is a good example in the December 1987 edition of *Your Commodore*. This is not essential though - you can make do with the commands and DISPLAY TAB on the demo disk you got with your drive.

Commands

The commands regarding direct disk access are called the Block commands (a Block is another name for a sector).



Your disk drive manual contains more detailed explanations, but a summary follows.

To use these commands, you'll need to have two files open - one for commands and the other to a buffer for data. The command channel you will probably be familiar with:

```
OPEN15,0,0
```

The data channel can be any other number, but 1 or 2 are usually used:

```
OPEN5,1,0
```

After these open commands, PRINT 15 will send commands, and PRINT 5 will send data to the channel.

Note that when 'drive' is mentioned, this means 0 for a single drive. The device is usually 0, but can be changed. See the examples on the disk for more information.

Block-Read

```
SYNTAX: PRINT #15, "B-R";
channel;
drive;
track;
sector;
```

This command transfers the required sector into the data channel (in our case 5). Then use the GET 5 command to read the information into a variable.

It's important to note that Block-Read will only read up as far as the Block-Pointer, which is usually 0. The USER1 command is usually used, as this sets the pointer to 255 automatically, allowing the sector to be read in one operation.

USER1

```
SYNTAX: PRINT #15, "U1";
channel;
drive;
track;
sector;
```

Block-Write

```
SYNTAX: PRINT #15, "B-W";
channel;
```

```
drive;
```

```
track;
```

```
sector;
```

To use this command, fill up the channel with information to write, using PRINT 5, then use the command to write to the required sector. This is the exact opposite of Block-Read, so again USER1 is usually used.

USER2

```
SYNTAX: PRINT #15, "U2";
channel;
```

```
drive;
track;
sector;
```

Block-Pointer

```
SYNTAX: PRINT #15, "B-P";
channel;
location;
```

By using this command, you can specify where exactly in the sector you want the next read or write to begin. This allows you to read or alter individual bytes in a sector, starting at 'location'. See the Disk Name program for a demo.

Block-Allocate

```
SYNTAX: PRINT #15, "B-A"; drive;
track;
```

This allocates a bit in the Block Availability Map to show a sector is in use. It is used in connection with random access databases.

Block-Free

```
SYNTAX: PRINT #15, "B-F"; drive;
track;
```

This is the opposite of Block-Allocate, and frees up sectors for use without destroying the actual data on them. If a name is made, the data will probably be overwritten, as the RAM has marked the sector as empty.

Disk Maps

Before you can use these commands, you will need some information on disk structure. The maps will provide this information, and information on file structure.

Editing

There are a number of files provided here for demonstration purposes. The best way to learn is to study these programs with the maps close at hand. They are heavily REBbed, but here are some notes explaining what's going on. Even if you don't know anything, they are useful utilities to have.

Protect File

This program 'locks' (i.e. prevents it from being scratched) the first file on a disk by setting bit 6 of the file type to 1, effectively OR'ing it with \$C0. This prevents accidental erasure, and has a < beside its name in the directory. By adding 32 to the last pointer number, and changing the sector number, any program in the directory may be protected.

Disk Name

This allows you to change the name of the disk without erasing the contents. It makes use of the fact that the disk name is stored at byte 144, track 18, sector 0.

Load Address

This changes the load address of any program to a given address. It searches for the first sector of storage, and bytes 2-3 contain the load address. It is most useful with spine data.

Unscrutch

On scratching a file, the filetype in the directory is manually marked as being deleted. This program searches the disk for a scratched program and restores the filetype, reconstructing the file. You are advised to save the unscratched program to another disk in case of another accident. Note that this will probably not work if something has been saved to the disk since the SCRATCH, as it may have been saved over the old program. Enjoy your disk editing!

BLOCK DISTRIBUTION BY TRACK

Track number	Block range	Total
1 to 17	0 to 30	21
18 to 24	0 to 18	19
25 to 30	0 to 17	18
31 to 33	0 to 16	17

1540/1541 BAM FORMAT

Track 18, Sector 0.		
BYTE	CONTENTS	DEFINITION
0,1	18,01	Track and block of first directory block.
2	43	ASCII character A indicating 4080 format.
3	0	Null flag for future DOS use.
4-143		Bit map of available blocks for tracks 1-35.
		*1 = available block 0 = block not available (each bit represents one block)

1540/1541 DIRECTORY HEADER

Track 18, Sector 0.		
BYTE	CONTENTS	DEFINITION
144-161		Disk name with shifted spaces.
162-163	160	Disk ID.
164		Shifted space
165-166	50,65	ASCII representation for 2A, which is DOS version and format type.
166-167	160	Shifted spaces.
177-255	0	Nulls, not used.
Note: ASCII characters may appear in locations 180 thru 191 on some diskettes.		

SEQUENTIAL FORMAT

BYTE	DEFINITION
0-1	Track and block of next sequential data block.
2-256	255 bytes of data with carriage return as record terminator.

PROGRAM FILE FORMAT

BYTE	DEFINITION
0-1	Track and block of next block in program file.
2-256	255 bytes of program info stored in CBM memory format (with key words colonised). End of file is marked by three zero bytes.

RELATIVE FILE FORMAT

DATA BLOCK	
BYTE	DEFINITION
0-1	Track and block of next data block.
2-256	254 bytes of data. Empty records contain FF (all binary ones) in the first byte followed by 99 (binary all ones) to the end of the record. Partially filled records are padded with nulls (00).
SIDE SECTOR BLOCK	
BYTE	DEFINITION
0-1	Track and block of next side sector block.
2	Side sector number (0-5).
3	Record length.
4-5	Track and block of first side sector (number 0).
6-7	Track and block of second side sector (number 1).
8-9	Track and block of third side sector (number 2).
10-11	Track and block of fourth side sector (number 3).
12-13	Track and block of fifth side sector (number 4).
14-15	Track and block of sixth side sector (number 5).
16-256	Track and block pointers to 120 data blocks.



WIN WITH COURSEMASTER

THE COMPUTER HORSE RACING PROGRAMME

- **PAYES ANY PRICE IN SECONDS** - ANY DAILY NEWSPAPER IS ALL YOU NEED
- **FASTER** and **cheaper** - Both in time and cost - Fast data entry
- **AMAZING ACCURACY** - Now you CAN BET THE **BOOKIES**
- **Works on the simple principle that BET COULD BEAT SLOW CHIEF**
- **Clearly identifies best selection in every race plus their AMAZING Features**
- **First, Second and Third place shown** for forecasts and Trebles etc. Forecasts most popular (up to 1 bet)
- **Actually works out your WINNINGS** of each popular bet including SINGLES AND DOUBLES, win and each way, PATENTS, TANGERS, CANADIAN, HEINIE etc. Good BACKWASH and LONG DOGS bets clearly shown
- **NEW! FIRST but also BETTING SLIP** for you
- **Maintains a BANK, ACCOUNT - BET LIKE PROFESSIONALS & Can** record all your bets in any number of accounts. Keep a complete record of your betting or compare CCM's results (70%) against your favorite tipster
- **Plus!** - The **AMAZING COURSEMASTER SYSTEM** - This sophisticated system is included in the programme. A system which regularly produces huge profits from small stakes. Try it out and pay for itself many times over on the top day!
- **Supplied with 36 page BETTING GUIDE AND MANUAL**
FREE HOT TIP OF THE MONTH SENT TO EVERY PURCHASER
Supplied on TAPE at £14.95 for -
All Spectrums, Commodore 64/
128, Amstrad CPC's, IBM &
Shogren Also available for
Sanyo CL on Microdrive at
£19.95.

Please state machine and day rate

**INTRASET LTD. (DEPT VC, FREEPOST (No Stamp
needed) 8 Gilderdale Close, Gorse Green, Bishwade,
Warrington, Cheshire, WA3 2BB. Or large SAE for
further details.**

PRINT ENHANCER Plus/4 disk £19.99

Power desktop publishing. Photo/lineart/expand. High quality printouts from BASIC, Fortran and spreadsheet. Supplied with 16 lines and four-page program. MFU-801/802, VIC 1200.

RS232 INTERFACE 64, 128 or Plus/4" ... £24.99

Measuring only 14x10x10cm, the smallest and lightest unit available. Conforms to Commodore 1514 user-wired standard. Compatible with BASIC, Easygraph, Supergraph, Superbase, View Office, Script Plus etc. Connect to RS232C printers, modems and other devices. Supplied with 1 metre of cable (500 The cable meter) terminated with a male (female) 25-way D-subminiature, or a custom cable made to your requirements. Our after sales service for unusual printers guaranteed to get you up and running. Supplied with a terminal emulator program, file transfer and other utilities on tape/disk

IBM PC File Transfer Utility £9.99

Captures text or binary files via the COM or RS232C port. Supplied on a 5.25" disk format 5 1/4" disk. Simple instructions provided by on-line help.

VIEWDATA TERMINAL 64 or Plus/4" ... £14.99

Access Probase, Microbase, CityDirector etc. Works with any user port RS232C interface and 1200/15 modem.

CENTRONICS CABLE 64, 128 £18.99

Use with Easygraph, Supergraph etc. Driver for BASICO on disk or tape - 6502 ASSEMBLER 64 or Plus/4" tape/disk £12.99

A sophisticated two-pass symbolic assembler and test editor which supports tape and disk filing. The assembler, editor, source and object code may all be resident in memory simultaneously, facilitating rapid and interactive code development. Very fast edit/assemble/test cycles.

280 EMULATOR/ASSEMBLER 64 disk £12.99

A unique integrated 280 development package. The emulator simulates 280 object code into assembled 6502 which runs on the 64 at about one sixth the speed of a 280/4 280. The cross-assembler generates hex or binary 280 object files. The disk contains a powerful editor and example programs.

Please specify options (and charge £5) or order by access.

Prices include VAT and P&H (overseas orders add £5.00).

Allow up to 1 week for delivery.

YORK ELECTRONIC RESEARCH

The Fitzhugh Centre, Dept VC, 4 Fitzhugh, York YO1 4AB
Tel (0904) 510120

100%

100%

A SUBSCRIPTION IS ONLY A CALL AWAY...

What could be a better way of keeping up to date with the latest news and developments in the world of the Commodore range of computers, than by ordering a direct subscription to 'YOUR COMMODORE', delivered direct to your door each month.

We've now made ordering a subscription easier than ever before by calling our CREDIT CARD HOTLINE. Simply give us a call, quoting your credit card details and delivery address and we'll do the rest! Remember, a subscription delivered to any address in the UK is POST FREE, all overseas subscriptions include postage.

Subscription Rates: UK £15.00, Europe £25.35, Middle East £28.50, Far East £26.40, Rest of the World £34.00 or USA \$42.00. (Airmail Rates on Request).



Telephone 0442 876661/4
Between usual office hours.



Audio Programs

[illegible][illegible][illegible]

1960	1961	1962	1963	1964	1965	1966	1967	1968	1969	1970	1971	1972	1973	1974	1975	1976	1977	1978	1979	1980	1981	1982	1983	1984	1985	1986	1987	1988	1989	1990	1991	1992	1993	1994	1995	1996	1997	1998	1999	2000	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	2012	2013	2014	2015	2016	2017	2018	2019	2020	2021	2022	2023	2024	2025	2026	2027	2028	2029	2030	2031	2032	2033	2034	2035	2036	2037	2038	2039	2040	2041	2042	2043	2044	2045	2046	2047	2048	2049	2050	2051	2052	2053	2054	2055	2056	2057	2058	2059	2060	2061	2062	2063	2064	2065	2066	2067	2068	2069	2070	2071	2072	2073	2074	2075	2076	2077	2078	2079	2080	2081	2082	2083	2084	2085	2086	2087	2088	2089	2090	2091	2092	2093	2094	2095	2096	2097	2098	2099	2100	2101	2102	2103	2104	2105	2106	2107	2108	2109	2110	2111	2112	2113	2114	2115	2116	2117	2118	2119	2120	2121	2122	2123	2124	2125	2126	2127	2128	2129	2130	2131	2132	2133	2134	2135	2136	2137	2138	2139	2140	2141	2142	2143	2144	2145	2146	2147	2148	2149	2150	2151	2152	2153	2154	2155	2156	2157	2158	2159	2160	2161	2162	2163	2164	2165	2166	2167	2168	2169	2170	2171	2172	2173	2174	2175	2176	2177	2178	2179	2180	2181	2182	2183	2184	2185	2186	2187	2188	2189	2190	2191	2192	2193	2194	2195	2196	2197	2198	2199	2200	2201	2202	2203	2204	2205	2206	2207	2208	2209	2210	2211	2212	2213	2214	2215	2216	2217	2218	2219	2220	2221	2222	2223	2224	2225	2226	2227	2228	2229	2230	2231	2232	2233	2234	2235	2236	2237	2238	2239	2240	2241	2242	2243	2244	2245	2246	2247	2248	2249	2250	2251	2252	2253	2254	2255	2256	2257	2258	2259	2260	2261	2262	2263	2264	2265	2266	2267	2268	2269	2270	2271	2272	2273	2274	2275	2276	2277	2278	2279	2280	2281	2282	2283	2284	2285	2286	2287	2288	2289	2290	2291	2292	2293	2294	2295	2296	2297	2298	2299	2300	2301	2302	2303	2304	2305	2306	2307	2308	2309	2310	2311	2312	2313	2314	2315	2316	2317	2318	2319	2320	2321	2322	2323	2324	2325	2326	2327	2328	2329	2330	2331	2332	2333	2334	2335	2336	2337	2338	2339	2340	2341	2342	2343	2344	2345	2346	2347	2348	2349	2350	2351	2352	2353	2354	2355	2356	2357	2358	2359	2360	2361	2362	2363	2364	2365	2366	2367	2368</
------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	--------

1. *Journal of the American Medical Association*, 1997; 277: 1039-1043.



```

00 10 1-01000
01 20 0000 1.07 0-000 THIS 0000
02 30 0000 1.07 0-000 THIS 0000
03 40 0000 1.07 0-000 THIS 0000
04 50 0000 1.07 0-000 THIS 0000
05 60 0000 1.07 0-000 THIS 0000
06 70 0000 1.07 0-000 THIS 0000
07 80 0000 1.07 0-000 THIS 0000
08 90 0000 1.07 0-000 THIS 0000
09 100 0000 1.07 0-000 THIS 0000
10 110 0000 1.07 0-000 THIS 0000
11 120 0000 1.07 0-000 THIS 0000
12 130 0000 1.07 0-000 THIS 0000
13 140 0000 1.07 0-000 THIS 0000
14 150 0000 1.07 0-000 THIS 0000
15 160 0000 1.07 0-000 THIS 0000
16 170 0000 1.07 0-000 THIS 0000
17 180 0000 1.07 0-000 THIS 0000
18 190 0000 1.07 0-000 THIS 0000
19 200 0000 1.07 0-000 THIS 0000
20 210 0000 1.07 0-000 THIS 0000
21 220 0000 1.07 0-000 THIS 0000
22 230 0000 1.07 0-000 THIS 0000
23 240 0000 1.07 0-000 THIS 0000
24 250 0000 1.07 0-000 THIS 0000
25 260 0000 1.07 0-000 THIS 0000
26 270 0000 1.07 0-000 THIS 0000
27 280 0000 1.07 0-000 THIS 0000
28 290 0000 1.07 0-000 THIS 0000
29 300 0000 1.07 0-000 THIS 0000
30 310 0000 1.07 0-000 THIS 0000
31 320 0000 1.07 0-000 THIS 0000
32 330 0000 1.07 0-000 THIS 0000
33 340 0000 1.07 0-000 THIS 0000
34 350 0000 1.07 0-000 THIS 0000
35 360 0000 1.07 0-000 THIS 0000
36 370 0000 1.07 0-000 THIS 0000
37 380 0000 1.07 0-000 THIS 0000
38 390 0000 1.07 0-000 THIS 0000
39 400 0000 1.07 0-000 THIS 0000
40 410 0000 1.07 0-000 THIS 0000
41 420 0000 1.07 0-000 THIS 0000
42 430 0000 1.07 0-000 THIS 0000
43 440 0000 1.07 0-000 THIS 0000
44 450 0000 1.07 0-000 THIS 0000
45 460 0000 1.07 0-000 THIS 0000
46 470 0000 1.07 0-000 THIS 0000
47 480 0000 1.07 0-000 THIS 0000
48 490 0000 1.07 0-000 THIS 0000
49 500 0000 1.07 0-000 THIS 0000
50 510 0000 1.07 0-000 THIS 0000
51 520 0000 1.07 0-000 THIS 0000
52 530 0000 1.07 0-000 THIS 0000
53 540 0000 1.07 0-000 THIS 0000
54 550 0000 1.07 0-000 THIS 0000
55 560 0000 1.07 0-000 THIS 0000
56 570 0000 1.07 0-000 THIS 0000
57 580 0000 1.07 0-000 THIS 0000
58 590 0000 1.07 0-000 THIS 0000
59 600 0000 1.07 0-000 THIS 0000
60 610 0000 1.07 0-000 THIS 0000
61 620 0000 1.07 0-000 THIS 0000
62 630 0000 1.07 0-000 THIS 0000
63 640 0000 1.07 0-000 THIS 0000
64 650 0000 1.07 0-000 THIS 0000
65 660 0000 1.07 0-000 THIS 0000
66 670 0000 1.07 0-000 THIS 0000
67 680 0000 1.07 0-000 THIS 0000
68 690 0000 1.07 0-000 THIS 0000
69 700 0000 1.07 0-000 THIS 0000
70 710 0000 1.07 0-000 THIS 0000
71 720 0000 1.07 0-000 THIS 0000
72 730 0000 1.07 0-000 THIS 0000
73 740 0000 1.07 0-000 THIS 0000
74 750 0000 1.07 0-000 THIS 0000
75 760 0000 1.07 0-000 THIS 0000
76 770 0000 1.07 0-000 THIS 0000
77 780 0000 1.07 0-000 THIS 0000
78 790 0000 1.07 0-000 THIS 0000
79 800 0000 1.07 0-000 THIS 0000
80 810 0000 1.07 0-000 THIS 0000
81 820 0000 1.07 0-000 THIS 0000
82 830 0000 1.07 0-000 THIS 0000
83 840 0000 1.07 0-000 THIS 0000
84 850 0000 1.07 0-000 THIS 0000
85 860 0000 1.07 0-000 THIS 0000
86 870 0000 1.07 0-000 THIS 0000
87 880 0000 1.07 0-000 THIS 0000
88 890 0000 1.07 0-000 THIS 0000
89 900 0000 1.07 0-000 THIS 0000
90 910 0000 1.07 0-000 THIS 0000
91 920 0000 1.07 0-000 THIS 0000
92 930 0000 1.07 0-000 THIS 0000
93 940 0000 1.07 0-000 THIS 0000
94 950 0000 1.07 0-000 THIS 0000
95 960 0000 1.07 0-000 THIS 0000
96 970 0000 1.07 0-000 THIS 0000
97 980 0000 1.07 0-000 THIS 0000
98 990 0000 1.07 0-000 THIS 0000
99 1000 0000 1.07 0-000 THIS 0000

```

84	000000	DATA	0.000	2.000	1.000	0.000
85	000000	DATA	0.000	1.000	0.000	0.000
86	000000	DATA	0.000	1.000	0.000	0.000
87	000000	DATA	0.000	1.000	0.000	0.000
88	000000	DATA	0.000	1.000	0.000	0.000
89	000000	DATA	0.000	1.000	0.000	0.000
90	000000	DATA	0.000	1.000	0.000	0.000
91	000000	DATA	0.000	1.000	0.000	0.000
92	000000	DATA	0.000	1.000	0.000	0.000
93	000000	DATA	0.000	1.000	0.000	0.000
94	000000	DATA	0.000	1.000	0.000	0.000
95	000000	DATA	0.000	1.000	0.000	0.000
96	000000	DATA	0.000	1.000	0.000	0.000
97	000000	DATA	0.000	1.000	0.000	0.000
98	000000	DATA	0.000	1.000	0.000	0.000
99	000000	DATA	0.000	1.000	0.000	0.000

[illegible]

Authors

[illegible]

10

10

LISTINGS

```

36 400 REM *** TEST ***
37 410 PRINT
38 420 IF (A=0) THEN GOTO 430
39 430 PRINT "A=0"
40 440 IF (A=1) THEN GOTO 450
41 450 PRINT "A=1"
42 460 IF (A=2) THEN GOTO 470
43 470 PRINT "A=2"
44 480 IF (A=3) THEN GOTO 490
45 490 PRINT "A=3"
46 500 IF (A=4) THEN GOTO 510
47 510 PRINT "A=4"
48 520 IF (A=5) THEN GOTO 530
49 530 PRINT "A=5"
50 540 IF (A=6) THEN GOTO 550
51 550 PRINT "A=6"
52 560 IF (A=7) THEN GOTO 570
53 570 PRINT "A=7"
54 580 IF (A=8) THEN GOTO 590
55 590 PRINT "A=8"
56 600 IF (A=9) THEN GOTO 610
57 610 PRINT "A=9"
58 620 IF (A=10) THEN GOTO 630
59 630 PRINT "A=10"
60 640 IF (A=11) THEN GOTO 650
61 650 PRINT "A=11"
62 660 IF (A=12) THEN GOTO 670
63 670 PRINT "A=12"
64 680 IF (A=13) THEN GOTO 690
65 690 PRINT "A=13"
66 700 IF (A=14) THEN GOTO 710
67 710 PRINT "A=14"
68 720 IF (A=15) THEN GOTO 730
69 730 PRINT "A=15"
70 740 IF (A=16) THEN GOTO 750
71 750 PRINT "A=16"
72 760 IF (A=17) THEN GOTO 770
73 770 PRINT "A=17"
74 780 IF (A=18) THEN GOTO 790
75 790 PRINT "A=18"
76 800 IF (A=19) THEN GOTO 810
77 810 PRINT "A=19"
78 820 IF (A=20) THEN GOTO 830
79 830 PRINT "A=20"
80 840 IF (A=21) THEN GOTO 850
81 850 PRINT "A=21"
82 860 IF (A=22) THEN GOTO 870
83 870 PRINT "A=22"
84 880 IF (A=23) THEN GOTO 890
85 890 PRINT "A=23"
86 900 IF (A=24) THEN GOTO 910
87 910 PRINT "A=24"
88 920 IF (A=25) THEN GOTO 930
89 930 PRINT "A=25"
90 940 IF (A=26) THEN GOTO 950
91 950 PRINT "A=26"
92 960 IF (A=27) THEN GOTO 970
93 970 PRINT "A=27"
94 980 IF (A=28) THEN GOTO 990
95 990 PRINT "A=28"
96 1000 IF (A=29) THEN GOTO 1010
97 1010 PRINT "A=29"
98 1020 IF (A=30) THEN GOTO 1030
99 1030 PRINT "A=30"
100 1040 IF (A=31) THEN GOTO 1050
101 1050 PRINT "A=31"
102 1060 IF (A=32) THEN GOTO 1070
103 1070 PRINT "A=32"
104 1080 IF (A=33) THEN GOTO 1090
105 1090 PRINT "A=33"
106 1100 IF (A=34) THEN GOTO 1110
107 1110 PRINT "A=34"
108 1120 IF (A=35) THEN GOTO 1130
109 1130 PRINT "A=35"
110 1140 IF (A=36) THEN GOTO 1150
111 1150 PRINT "A=36"
112 1160 IF (A=37) THEN GOTO 1170
113 1170 PRINT "A=37"
114 1180 IF (A=38) THEN GOTO 1190
115 1190 PRINT "A=38"
116 1200 IF (A=39) THEN GOTO 1210
117 1210 PRINT "A=39"
118 1220 IF (A=40) THEN GOTO 1230
119 1230 PRINT "A=40"
120 1240 IF (A=41) THEN GOTO 1250
121 1250 PRINT "A=41"
122 1260 IF (A=42) THEN GOTO 1270
123 1270 PRINT "A=42"
124 1280 IF (A=43) THEN GOTO 1290
125 1290 PRINT "A=43"
126 1300 IF (A=44) THEN GOTO 1310
127 1310 PRINT "A=44"
128 1320 IF (A=45) THEN GOTO 1330
129 1330 PRINT "A=45"
130 1340 IF (A=46) THEN GOTO 1350
131 1350 PRINT "A=46"
132 1360 IF (A=47) THEN GOTO 1370
133 1370 PRINT "A=47"
134 1380 IF (A=48) THEN GOTO 1390
135 1390 PRINT "A=48"
136 1400 IF (A=49) THEN GOTO 1410
137 1410 PRINT "A=49"
138 1420 IF (A=50) THEN GOTO 1430
139 1430 PRINT "A=50"
140 1440 IF (A=51) THEN GOTO 1450
141 1450 PRINT "A=51"
142 1460 IF (A=52) THEN GOTO 1470
143 1470 PRINT "A=52"
144 1480 IF (A=53) THEN GOTO 1490
145 1490 PRINT "A=53"
146 1500 IF (A=54) THEN GOTO 1510
147 1510 PRINT "A=54"
148 1520 IF (A=55) THEN GOTO 1530
149 1530 PRINT "A=55"
150 1540 IF (A=56) THEN GOTO 1550
151 1550 PRINT "A=56"
152 1560 IF (A=57) THEN GOTO 1570
153 1570 PRINT "A=57"
154 1580 IF (A=58) THEN GOTO 1590
155 1590 PRINT "A=58"
156 1600 IF (A=59) THEN GOTO 1610
157 1610 PRINT "A=59"
158 1620 IF (A=60) THEN GOTO 1630
159 1630 PRINT "A=60"
160 1640 IF (A=61) THEN GOTO 1650
161 1650 PRINT "A=61"
162 1660 IF (A=62) THEN GOTO 1670
163 1670 PRINT "A=62"
164 1680 IF (A=63) THEN GOTO 1690
165 1690 PRINT "A=63"
166 1700 IF (A=64) THEN GOTO 1710
167 1710 PRINT "A=64"
168 1720 IF (A=65) THEN GOTO 1730
169 1730 PRINT "A=65"
170 1740 IF (A=66) THEN GOTO 1750
171 1750 PRINT "A=66"
172 1760 IF (A=67) THEN GOTO 1770
173 1770 PRINT "A=67"
174 1780 IF (A=68) THEN GOTO 1790
175 1790 PRINT "A=68"
176 1800 IF (A=69) THEN GOTO 1810
177 1810 PRINT "A=69"
178 1820 IF (A=70) THEN GOTO 1830
179 1830 PRINT "A=70"
180 1840 IF (A=71) THEN GOTO 1850
181 1850 PRINT "A=71"
182 1860 IF (A=72) THEN GOTO 1870
183 1870 PRINT "A=72"
184 1880 IF (A=73) THEN GOTO 1890
185 1890 PRINT "A=73"
186 1900 IF (A=74) THEN GOTO 1910
187 1910 PRINT "A=74"
188 1920 IF (A=75) THEN GOTO 1930
189 1930 PRINT "A=75"
190 1940 IF (A=76) THEN GOTO 1950
191 1950 PRINT "A=76"
192 1960 IF (A=77) THEN GOTO 1970
193 1970 PRINT "A=77"
194 1980 IF (A=78) THEN GOTO 1990
195 1990 PRINT "A=78"
196 2000 IF (A=79) THEN GOTO 2010
197 2010 PRINT "A=79"
198 2020 IF (A=80) THEN GOTO 2030
199 2030 PRINT "A=80"
200 2040 IF (A=81) THEN GOTO 2050
201 2050 PRINT "A=81"
202 2060 IF (A=82) THEN GOTO 2070
203 2070 PRINT "A=82"
204 2080 IF (A=83) THEN GOTO 2090
205 2090 PRINT "A=83"
206 2100 IF (A=84) THEN GOTO 2110
207 2110 PRINT "A=84"
208 2120 IF (A=85) THEN GOTO 2130
209 2130 PRINT "A=85"
210 2140 IF (A=86) THEN GOTO 2150
211 2150 PRINT "A=86"
212 2160 IF (A=87) THEN GOTO 2170
213 2170 PRINT "A=87"
214 2180 IF (A=88) THEN GOTO 2190
215 2190 PRINT "A=88"
216 2200 IF (A=89) THEN GOTO 2210
217 2210 PRINT "A=89"
218 2220 IF (A=90) THEN GOTO 2230
219 2230 PRINT "A=90"
220 2240 IF (A=91) THEN GOTO 2250
221 2250 PRINT "A=91"
222 2260 IF (A=92) THEN GOTO 2270
223 2270 PRINT "A=92"
224 2280 IF (A=93) THEN GOTO 2290
225 2290 PRINT "A=93"
226 2300 IF (A=94) THEN GOTO 2310
227 2310 PRINT "A=94"
228 2320 IF (A=95) THEN GOTO 2330
229 2330 PRINT "A=95"
230 2340 IF (A=96) THEN GOTO 2350
231 2350 PRINT "A=96"
232 2360 IF (A=97) THEN GOTO 2370
233 2370 PRINT "A=97"
234 2380 IF (A=98) THEN GOTO 2390
235 2390 PRINT "A=98"
236 2400 IF (A=99) THEN GOTO 2410
237 2410 PRINT "A=99"
238 2420 IF (A=100) THEN GOTO 2430
239 2430 PRINT "A=100"
240 2440 IF (A=101) THEN GOTO 2450
241 2450 PRINT "A=101"
242 2460 IF (A=102) THEN GOTO 2470
243 2470 PRINT "A=102"
244 2480 IF (A=103) THEN GOTO 2490
245 2490 PRINT "A=103"
246 2500 IF (A=104) THEN GOTO 2510
247 2510 PRINT "A=104"
248 2520 IF (A=105) THEN GOTO 2530
249 2530 PRINT "A=105"
250 2540 IF (A=106) THEN GOTO 2550
251 2550 PRINT "A=106"
252 2560 IF (A=107) THEN GOTO 2570
253 2570 PRINT "A=107"
254 2580 IF (A=108) THEN GOTO 2590
255 2590 PRINT "A=108"
256 2600 IF (A=109) THEN GOTO 2610
257 2610 PRINT "A=109"
258 2620 IF (A=110) THEN GOTO 2630
259 2630 PRINT "A=110"
260 2640 IF (A=111) THEN GOTO 2650
261 2650 PRINT "A=111"
262 2660 IF (A=112) THEN GOTO 2670
263 2670 PRINT "A=112"
264 2680 IF (A=113) THEN GOTO 2690
265 2690 PRINT "A=113"
266 2700 IF (A=114) THEN GOTO 2710
267 2710 PRINT "A=114"
268 2720 IF (A=115) THEN GOTO 2730
269 2730 PRINT "A=115"
270 2740 IF (A=116) THEN GOTO 2750
271 2750 PRINT "A=116"
272 2760 IF (A=117) THEN GOTO 2770
273 2770 PRINT "A=117"
274 2780 IF (A=118) THEN GOTO 2790
275 2790 PRINT "A=118"
276 2800 IF (A=119) THEN GOTO 2810
277 2810 PRINT "A=119"
278 2820 IF (A=120) THEN GOTO 2830
279 2830 PRINT "A=120"
280 2840 IF (A=121) THEN GOTO 2850
281 2850 PRINT "A=121"
282 2860 IF (A=122) THEN GOTO 2870
283 2870 PRINT "A=122"
284 2880 IF (A=123) THEN GOTO 2890
285 2890 PRINT "A=123"
286 2900 IF (A=124) THEN GOTO 2910
287 2910 PRINT "A=124"
288 2920 IF (A=125) THEN GOTO 2930
289 2930 PRINT "A=125"
290 2940 IF (A=126) THEN GOTO 2950
291 2950 PRINT "A=126"
292 2960 IF (A=127) THEN GOTO 2970
293 2970 PRINT "A=127"
294 2980 IF (A=128) THEN GOTO 2990
295 2990 PRINT "A=128"
296 3000 IF (A=129) THEN GOTO 3010
297 3010 PRINT "A=129"
298 3020 IF (A=130) THEN GOTO 3030
299 3030 PRINT "A=130"
300 3040 IF (A=131) THEN GOTO 3050
301 3050 PRINT "A=131"
302 3060 IF (A=132) THEN GOTO 3070
303 3070 PRINT "A=132"
304 3080 IF (A=133) THEN GOTO 3090
305 3090 PRINT "A=133"
306 3100 IF (A=134) THEN GOTO 3110
307 3110 PRINT "A=134"
308 3120 IF (A=135) THEN GOTO 3130
309 3130 PRINT "A=135"
310 3140 IF (A=136) THEN GOTO 3150
311 3150 PRINT "A=136"
312 3160 IF (A=137) THEN GOTO 3170
313 3170 PRINT "A=137"
314 3180 IF (A=138) THEN GOTO 3190
315 3190 PRINT "A=138"
316 3200 IF (A=139) THEN GOTO 3210
317 3210 PRINT "A=139"
318 3220 IF (A=140) THEN GOTO 3230
319 3230 PRINT "A=140"
320 3240 IF (A=141) THEN GOTO 3250
321 3250 PRINT "A=141"
322 3260 IF (A=142) THEN GOTO 3270
323 3270 PRINT "A=142"
324 3280 IF (A=143) THEN GOTO 3290
325 3290 PRINT "A=143"
326 3300 IF (A=144) THEN GOTO 3310
327 3310 PRINT "A=144"
328 3320 IF (A=145) THEN GOTO 3330
329 3330 PRINT "A=145"
330 3340 IF (A=146) THEN GOTO 3350
331 3350 PRINT "A=146"
332 3360 IF (A=147) THEN GOTO 3370
333 3370 PRINT "A=147"
334 3380 IF (A=148) THEN GOTO 3390
335 3390 PRINT "A=148"
336 3400 IF (A=149) THEN GOTO 3410
337 3410 PRINT "A=149"
338 3420 IF (A=150) THEN GOTO 3430
339 3430 PRINT "A=150"
340 3440 IF (A=151) THEN GOTO 3450
341 3450 PRINT "A=151"
342 3460 IF (A=152) THEN GOTO 3470
343 3470 PRINT "A=152"
344 3480 IF (A=153) THEN GOTO 3490
345 3490 PRINT "A=153"
346 3500 IF (A=154) THEN GOTO 3510
347 3510 PRINT "A=154"
348 3520 IF (A=155) THEN GOTO 3530
349 3530 PRINT "A=155"
350 3540 IF (A=156) THEN GOTO 3550
351 3550 PRINT "A=156"
352 3560 IF (A=157) THEN GOTO 3570
353 3570 PRINT "A=157"
354 3580 IF (A=158) THEN GOTO 3590
355 3590 PRINT "A=158"
356 3600 IF (A=159) THEN GOTO 3610
357 3610 PRINT "A=159"
358 3620 IF (A=160) THEN GOTO 3630
359 3630 PRINT "A=160"
360 3640 IF (A=161) THEN GOTO 3650
361 3650 PRINT "A=161"
362 3660 IF (A=162) THEN GOTO 3670
363 3670 PRINT "A=162"
364 3680 IF (A=163) THEN GOTO 3690
365 3690 PRINT "A=163"
366 3700 IF (A=164) THEN GOTO 3710
367 3710 PRINT "A=164"
368 3720 IF (A=165) THEN GOTO 3730
369 3730 PRINT "A=165"
370 3740 IF (A=166) THEN GOTO 3750
371 3750 PRINT "A=166"
372 3760 IF (A=167) THEN GOTO 3770
373 3770 PRINT "A=167"
374 3780 IF (A=168) THEN GOTO 3790
375 3790 PRINT "A=168"
376 3800 IF (A=169) THEN GOTO 3810
377 3810 PRINT "A=169"
378 3820 IF (A=170) THEN GOTO 3830
379 3830 PRINT "A=170"
380 3840 IF (A=171) THEN GOTO 3850
381 3850 PRINT "A=171"
382 3860 IF (A=172) THEN GOTO 3870
383 3870 PRINT "A=172"
384 3880 IF (A=173) THEN GOTO 3890
385 3890 PRINT "A=173"
386 3900 IF (A=174) THEN GOTO 3910
387 3910 PRINT "A=174"
388 3920 IF (A=175) THEN GOTO 3930
389 3930 PRINT "A=175"
390 3940 IF (A=176) THEN GOTO 3950
391 3950 PRINT "A=176"
392 3960 IF (A=177) THEN GOTO 3970
393 3970 PRINT "A=177"
394 3980 IF (A=178) THEN GOTO 3990
395 3990 PRINT "A=178"
396 4000 IF (A=179) THEN GOTO 4010
397 4010 PRINT "A=179"
398 4020 IF (A=180) THEN GOTO 4030
399 4030 PRINT "A=180"
400 4040 IF (A=181) THEN GOTO 4050
401 4050 PRINT "A=181"
402 4060 IF (A=182) THEN GOTO 4070
403 4070 PRINT "A=182"
404 4080 IF (A=183) THEN GOTO 4090
405 4090 PRINT "A=183"
406 4100 IF (A=184) THEN GOTO 4110
407 4110 PRINT "A=184"
408 4120 IF (A=185) THEN GOTO 4130
409 4130 PRINT "A=185"
410 4140 IF (A=186) THEN GOTO 4150
411 4150 PRINT "A=186"
412 4160 IF (A=187) THEN GOTO 4170
413 4170 PRINT "A=187"
414 4180 IF (A=188) THEN GOTO 4190
415 4190 PRINT "A=188"
416 4200 IF (A=189) THEN GOTO 4210
417 4210 PRINT "A=189"
418 4220 IF (A=190) THEN GOTO 4230
419 4230 PRINT "A=190"
420 4240 IF (A=191) THEN GOTO 4250
421 4250 PRINT "A=191"
422 4260 IF (A=192) THEN GOTO 4270
423 4270 PRINT "A=192"
424 4280 IF (A=193) THEN GOTO 4290
425 4290 PRINT "A=193"
426 4300 IF (A=194) THEN GOTO 4310
427 4310 PRINT "A=194"
428 4320 IF (A=195) THEN GOTO 4330
429 4330 PRINT "A=195"
430 4340 IF (A=196) THEN GOTO 4350
431 4350 PRINT "A=196"
432 4360 IF (A=197) THEN GOTO 4370
433 4370 PRINT "A=197"
434 4380 IF (A=198) THEN GOTO 4390
435 4390 PRINT "A=198"
436 4400 IF (A=199) THEN GOTO 4410
437 4410 PRINT "A=199"
438 4420 IF (A=200) THEN GOTO 4430
439 4430 PRINT "A=200"
440 4440 IF (A=201) THEN GOTO 4450
441 4450 PRINT "A=201"
442 4460 IF (A=202) THEN GOTO 4470
443 4470 PRINT "A=202"
444 4480 IF (A=203) THEN GOTO 4490
445 4490 PRINT "A=203"
446 4500 IF (A=204) THEN GOTO 4510
447 4510 PRINT "A=204"
448 4520 IF (A=205) THEN GOTO 4530
449 4530 PRINT "A=205"
450 4540 IF (A=206) THEN GOTO 4550
451 4550 PRINT "A=206"
452 4560 IF (A=207) THEN GOTO 4570
453 4570 PRINT "A=207"
454 4580 IF (A=208) THEN GOTO 4590
455 4590 PRINT "A=208"
456 4600 IF (A=209) THEN GOTO 4610
457 4610 PRINT "A=209"
458 4620 IF (A=210) THEN GOTO 4630
459 4630 PRINT "A=210"
460 4640 IF (A=211) THEN GOTO 4650
461 4650 PRINT "A=211"
462 4660 IF (A=212) THEN GOTO 4670
463 4670 PRINT "A=212"
464 4680 IF (A=213) THEN GOTO 4690
465 4690 PRINT "A=213"
466 4700 IF (A=214) THEN GOTO 4710
467 4710 PRINT "A=214"
468 4720 IF (A=215) THEN GOTO 4730
469 4730 PRINT "A=215"
470 4740 IF (A=216) THEN GOTO 4750
471 4750 PRINT "A=216"
472 4760 IF (A=217) THEN GOTO 4770
473 4770 PRINT "A=217"
474 4780 IF (A=218) THEN GOTO 4790
475 4790 PRINT "A=218"
476 4800 IF (A=219) THEN GOTO 4810
477 4810 PRINT "A=219"
478 4820 IF (A=220) THEN GOTO 4830
479 4830 PRINT "A=220"
480 4840 IF (A=221) THEN GOTO 4850
481 4850 PRINT "A=221"
482 4860 IF (A=222) THEN GOTO 4870
483 4870 PRINT "A=222"
484 4880 IF (A=223) THEN GOTO 4890
485 4890 PRINT "A=223"
486 4900 IF (A=224) THEN GOTO 4910
487 4910 PRINT "A=224"
488 4920 IF (A=225) THEN GOTO 4930
489 4930 PRINT "A=225"
490 4940 IF (A=226) THEN GOTO 4950
491 4950 PRINT "A=226"
492 4960 IF (A=227) THEN GOTO 4970
493 4970 PRINT "A=227"
494 4980 IF (A=228) THEN GOTO 4990
495 4990 PRINT "A=228"
496 5000 IF (A=229) THEN GOTO 5010
497 5010 PRINT "A=229"
498 5020 IF (A=230) THEN GOTO 5030
499 5030 PRINT "A=230"
500 5040 IF (A=231) THEN GOTO 5050
501 5050 PRINT "A=231"
502 5060 IF (A=232) THEN GOTO 5070
503 5070 PRINT "A=232"
504 5080 IF (A=233) THEN GOTO 5090
505 5090 PRINT "A=233"
506 5100 IF (A=234) THEN GOTO 5110
507 5110 PRINT "A=234"
508 5120 IF (A=235) THEN GOTO 5130
509 5130 PRINT "A=235"
510 5140 IF (A=236) THEN GOTO 5150
511 5150 PRINT "A=236"
512 5160 IF (A=237) THEN GOTO 5170
513 5170 PRINT "A=237"
514 5180 IF (A=238) THEN GOTO 5190
515 5190 PRINT "A=238"
516 5200 IF (A=239) THEN GOTO 5210
517 5210 PRINT "A=239"
518 5220 IF (A=240) THEN GOTO 5230
519 5230 PRINT "A=240"
520 5240 IF (A=241) THEN GOTO 5250
521 5250 PRINT "A=241"
522 5260 IF (A=242) THEN GOTO 5270
523 5270 PRINT "A=242"
524 5280 IF (A=243) THEN GOTO 5290
525 5290 PRINT "A=243"
526 5300 IF (A=244) THEN GOTO 5310
527 5310 PRINT "A=244"
528 5320 IF (A=245) THEN GOTO 5330
529 5330 PRINT "A=245"
530 5340 IF (A=246) THEN GOTO 5350
531 5350 PRINT "A=246"
532 5360 IF (A=247) THEN GOTO 5370
533 5370 PRINT "A=247"
534 5380 IF (A=248) THEN GOTO 5390
535 5390 PRINT "A=248"
536 5400 IF (A=249) THEN GOTO 5410
537 5410 PRINT "A=249"
538 5420 IF (A=250) THEN GOTO 5430
539 5430 PRINT "A=250"
540 5440 IF (A=251) THEN GOTO 5450
541 5450 PRINT "A=251"
542 5460 IF (A=252) THEN GOTO 5470
543 5470 PRINT "A=252"
544 5480 IF (A=253) THEN GOTO 5490
545 5490 PRINT "A=253"
546 5500 IF (A=254) THEN GOTO 5510
547 5510 PRINT "A=254"
548 5520 IF (A=255) THEN GOTO 5530
549 5530 PRINT "A=255"
550 5540 IF (A=256) THEN GOTO 5550
551 5550 PRINT "A=256"
552 5560 IF (A=257) THEN GOTO 5570
553 5570 PRINT "A=257"
554 5580 IF (A=258) THEN GOTO 5590
555 5590 PRINT "A=258"
556 5600 IF (A=259) THEN GOTO 5610
557 5610 PRINT "A=259"
558 5620 IF (A=260) THEN GOTO 5630
559 5630 PRINT "A=260"
560 5640 IF (A=261) THEN GOTO 5650
561 5650 PRINT "A=261"
562 5660 IF (A=262) THEN GOTO 5670
563 5670 PRINT "A=262"
564 5680 IF (A=263) THEN GOTO 5690
565 5690 PRINT "A=263"
566 5700 IF (A=264) THEN GOTO 5710
567 5710 PRINT "A=264"
568 5720 IF (A=265) THEN GOTO 5730
569 5730 PRINT "A=265"
570 5740 IF (A=266) THEN GOTO 5750
571 5750 PRINT "A=266"
572 5760 IF (A=267) THEN GOTO 5770
573 5770 PRINT "A=267"
574 5780 IF (A=268) THEN GOTO 5790
575 5790 PRINT "A=268"
576 5800 IF (A=269) THEN GOTO 5810
577 5810 PRINT "A=269"
578 5820 IF (A=270) THEN GOTO 5830
579 5830 PRINT "A=270"
580 5840 IF (A=271) THEN GOTO 5850
581 5850 PRINT "A=271"
582 5860 IF (A=272) THEN GOTO 5870
583 5870 PRINT "A=272"
584 5880 IF (A=273) THEN GOTO 5890
585 5890 PRINT "A=273"
586 5900 IF (A=274) THEN GOTO 5910
587 5910 PRINT "A=274"
588 5920 IF (A=275) THEN GOTO 5930
589 5930 PRINT "A=275"
590 5940 IF (A=276) THEN GOTO 5950
591 5950 PRINT "A=276"
592 5960 IF (A=277) THEN GOTO 5970
593 5970 PRINT "A=277"
594 5980 IF (A=278) THEN GOTO 5990
595 5990 PRINT "A=278"
596 6000 IF (A=279) THEN GOTO 6010
597 6010 PRINT "A=279"
598 6020 IF (A=280) THEN GOTO 6030
599 6030 PRINT "A=280"
600 6040 IF (A=281) THEN GOTO 6050
601 6050 PRINT "A=281"
602 6060 IF (A=282) THEN GOTO 6070
603 6070 PRINT "A=282"
604 6080 IF (A=283) THEN GOTO 6090
605 6090 PRINT "A=283"
606 6100 IF (A=284) THEN GOTO 6110
607 6110 PRINT "A=284"
608 6120 IF (A=285) THEN GOTO 6130
609 6130 PRINT "A=285"
610 6140 IF (A=286) THEN GOTO 6150
611 6150 PRINT "A=286"
612 6160 IF (A=287) THEN GOTO 6170
613 6170 PRINT "A=287"
614 6180 IF (A=288) THEN GOTO 6190
615 6190 PRINT "A=288"
616 6200 IF (A=289) THEN GOTO 6210
617 6210 PRINT "A=289"
618 6220 IF (A=290) THEN GOTO 6230
619 6230 PRINT "A=290"
620 6240 IF (A=291) THEN GOTO 6250
621 6250 PRINT "A=291"
622 6260 IF (A=292) THEN GOTO 6270
623 6270 PRINT "A=292"
624 6280 IF (A=293) THEN GOTO 6290
625 6290 PRINT "A=293"
626 6300 IF (A=294) THEN GOTO 6310
627 6310 PRINT "A=294"
628 6320 IF (A=295) THEN GOTO 6330
629 6330 PRINT "A=295"
630 6340 IF (A=296) THEN GOTO 6350
631 6350 PRINT "A=296"
632 6360 IF (A=297) THEN GOTO 6370
633 6370 PRINT "A=297"
634 6380 IF (A=298) THEN GOTO 6390
635 6390 PRINT "A=298"
636 6400 IF (A=299) THEN GOTO 6410
637 6410 PRINT "A=299"
638 6420 IF (A=300) THEN GOTO 6430
639 6430 PRINT "A=300"
640 6440 IF (A=301) THEN GOTO 645
```


LISTINGS

83	21144	IFRANCE	14-PTRENA-BOC	14-15
84	21145	IFRANCE	14-PTRENA-BOC	14-15
85	21146	IFRANCE	14-PTRENA-BOC	14-15
86	21147	IFRANCE	14-PTRENA-BOC	14-15
87	21148	IFRANCE	14-PTRENA-BOC	14-15
88	21149	IFRANCE	14-PTRENA-BOC	14-15
89	21150	IFRANCE	14-PTRENA-BOC	14-15
90	21151	IFRANCE	14-PTRENA-BOC	14-15
91	21152	IFRANCE	14-PTRENA-BOC	14-15
92	21153	IFRANCE	14-PTRENA-BOC	14-15
93	21154	IFRANCE	14-PTRENA-BOC	14-15
94	21155	IFRANCE	14-PTRENA-BOC	14-15
95	21156	IFRANCE	14-PTRENA-BOC	14-15
96	21157	IFRANCE	14-PTRENA-BOC	14-15
97	21158	IFRANCE	14-PTRENA-BOC	14-15
98	21159	IFRANCE	14-PTRENA-BOC	14-15
99	21160	IFRANCE	14-PTRENA-BOC	14-15

Thompson and Thompson



Abstract

[illegible][illegible]

```

      EN,3=1"
04      DOO PRINT*IDENR,B-,C7,C8,C9,
AL," ",FORM=2)FORM=PRINT*IDENR,
",",B7,B8,B9,C6,C7,C8,C9,
"/",C7,C8,C9,
06      DOO PRINT*IDENR,A;NEXT=PRINT*
IDENR,IDENR,B-3"
08      DOO PRINT*IDENR,C7,C8,C9,END"

```

Response	Percentage
U.S. should take action	55%
U.S. should not take action	45%

[illegible]

10

LISTINGS

[illegible]

[Home](#) / [About Us](#) / [Contact Us](#) / [Privacy Policy](#) / [Terms & Conditions](#)

100

Binders

Why not keep your Commodore Disk User program collection alongside your magazines in a stylish Disk User binder? The binder comes complete with 10 disk sleeves to organize and protect your program disks. Why not buy a disk binder to house all of your disk disks? We can even supply Commodore Disk User disk disks. The Commodore Disk User logs immediately identifies your disks and there's room to note them and document the disks details.

Send for your disks and binder now!

All orders should be sent to: YOUR COMMODORE, READERS SERVICES,
ARGUS SPECIALIST PUBLICATIONS, 9 HALL ROAD, HEMEL HEMPSTEAD,
HERTS HP27 7BN. Please allow 28 days for delivery.

Extending Basic

Declaring and using labels in Basic is not as difficult as it may seem

By Burghard-Heinz Lehmann

One advantage of writing machine code programs with an assembler is that you can declare labels for jumps and branches. An example of this is "Speedy Assembler", written by yours truly, and still available from Readers' Service for everybody who wants to go into serious machine code programming!

Instead of having to give jumps and subroutine calls as absolute addresses, and having to calculate relative branches, you simply declare a label on the left-hand side at the start of the routine to be jumped or branched to, and the assembler does the rest. It stores the label in what is called "the symbol table", with the address of the location at that point next to it.

Later on, when it finds that label next to a jump or branch instruction, it fetches the address from the symbol table and, in case of a jump or subroutine call, assembles it in the location to be jumped or to be called on, in the case of a branch instruction, calculates the length of the branch. The programmer doesn't have to trouble himself with any of this. Nor does he have to go to the trouble of calculating branches. Instead he just attaches a name to the routine or subroutine in question.

And what's more, labels like this add tremendously to the readability of the program, because you can give each routine and subroutine a name that suits you. This helps quite a bit when a bug has developed in the

program (and doesn't it always!), and the programmer has to spend ages finding it!

Labels in Basic

Because of all this, I felt for a long time that it would be nice to be able to use labels like this in Basic programs too. No more memorizing of line numbers, and, most of all, no more remembering of GOTO's and GOSUB's whenever one changes the program.

It is actually surprisingly easy to introduce such a facility to the rather poverty-stricken Commodore 64 Basic. In the last article of this series we've developed a routine which allows us to give GOTO and GOSUB with variable names. So we've already got the basic facility to labelize jumps and subroutine calls, but we still have to declare the value of the label at the beginning of the program with a line

like "Subroutine = 1000". To do this job for us, the computer has to build a symbol table of sorts, and this has to be done before the program is actually run. This is because, during execution, when a GOTO or GOSUB is encountered, the computer has to know where to jump to.

You may know that most assemblers are called "Two Pass Assemblers". This is because in order to deal with jumps and branches, the assembler has to do its job in two goes. First it goes through the whole of the textfile and builds the symbol table, and then it has a second go in which it is able to assemble the source in correct.

To use labels to the full in Basic, we have to do a similar thing. Before the program is run in earnest, the computer has to sift through the whole of the textfile and collect all the line numbers to which it has to GOTO and GOSUB later on. This means that it

Figure 1

A871	BasicRun	PHP	Save status register
A872		LDA #	This is direct run mode
A874		JSR \$FF90	Set Run/Alt flags to direct run mode
A871		PLP	Retrieve status register
A876		BNE LineNo	If Z=0 then RUN plus line number
A87A		JMP	Do CLR and start program
A87D	LineNo	JSR \$A660	Do CLR
A880		JMP \$A697	Jump to GOTO and start program

takes a bit of time until running proper starts. But I don't think that this is a major handicap, and is certainly is worth it!

Running a Basic Program

So we have to interrupt the RUN command. To do this, let's look first of all a bit closer at what it does under normal circumstances.

After you've typed in your Basic program and then given the RUN command to execute your program, the Basic interpreter jumps to the routine at \$A821. Figure 1 gives you a disassembly of that routine.

First the flag register is saved on the stack. Then a call to a Kernel routine is made. This routine puts the computer into the direct run mode by loading the system variable \$RD with zero and setting ST, which is the status variable.

Then the flag register is pulled from the stack again. If the zero flag equals one, that is, if the last value in the accumulator has been zero, then there were no parameters with the RUN command. In this case the routine continues, otherwise it branches forward, because a line number has been given with RUN.

If no line number is given with RUN the routine jumps to the CLR routine and doesn't return. If a line number has been given with RUN, the routine calls on the CLR routine and then jumps to the GOTO routine, because in the real, RUN 100 is similar to GOTO 100. The only difference is that RUN clears all the Basic variables, while GOTO leaves them untouched.

The CLR routine clears all the Basic variables and gives Basic a fresh start, so that variables and arrays can be built up anew. This is mainly done by setting the string storage pointer and the array storage pointer to the end of the Basic textfile, because, as you might know, Basic stores all the variables and arrays declared in a program directly after the end of the textfile.

Modified RUN

Our modified RUN routine starts at line 1326 (Listing 1). First we deal with the RUN command more or less in the same way as the ROM routine. That is, we set the Kernel flag to the direct run mode (line 1328-1330), and then we clear the Basic variables (line 1340).

I haven't bothered about RUN line number, so this won't work with the routine as it is. If you give a line number with RUN, it will just be ignored. But nevertheless, if you do want this little-used facility, it shouldn't be impossible for you to add it with the help of the explanation of the normal RUN routine, which I've given above.

Next, we set zero page 234/235 to the start of the Basic textfile (line 1348-1410). Zero page locations 231 to 235 are never used by the Commodore operating system, so these locations are absolutely safe to be used for your own purposes. There are many other zero page locations which are used by the Kernel or the Basic interpreter, but which are usually quite safe to use. For example, if you don't do any floating point arithmetic, you may use locations \$64 to \$6F without any trouble. But the point is always to think before you use a zero page variable, otherwise the system might do very funny things indeed!

After that we go into the main loop (SEARCHLIP), which looks at each Basic line to see if it contains a label (line 1636-1610). To understand SEARCHLIP, here is a short explanation of how a Basic line is stored in memory:

First of all the line number is given in the usual low byte/high byte fashion. Next there are two bytes which contain the so-called link pointer. Each link pointer points to the beginning of the next line. This makes it very easy to search through a Basic textfile, because you just have to jump from one link pointer to the other, each time looking at the line number preceding it, and in next to no time you've found the line you're looking for.

Each Basic line is finished with a zero, which is the standard terminator used by Commodore (strings too are always terminated with zero). At the end of the textfile there will be two zeros in the locations where otherwise the next line number would be. So this is how the computer will know when it has reached the end of the textfile.

If you look at SEARCHLIP in our program, you'll find the routine testing for those two zeros right at the beginning of the loop (line 1636-1660).

Declaring a Label

Naturally we have to tell the computer when it has found a label. For that

we have to make a label stand out in some way.

To do this I have chosen the following way of declaring a label: a label has to be at the beginning of a line, after the line number, and it has to be preceded by a full stop. Of course, you are free to experiment with methods which might suit you better, because this is the whole purpose of this series of articles, to enable you to develop extended Basic routines which suit your particular needs!

Anyhow, in the routine given, the computer looks for a full stop and that tells him that it has found a label (line 1666-1720). So it jumps to the routine which I called LABELFOUND. First of all the current location in the textfile is stored in STA/TH, so that afterwards the computer can continue searching the textfile for more labels (line 1990-1998).

Then it goes forward five bytes to point at the label itself (line 2000-2004). Remember, all we are doing here is a simple LFT operation, i.e. "LET LABEL = 1608" (or "LABEL = 1608" if you omit the LET). So the next ROM routine (\$B080) we call validates our label, that is, finds out if it is a permitted variable name (as you know, a valid Basic variable name has to start with a letter).

If the name is valid, the first two characters of it are stored in the variable area, which starts immediately after the end of the Basic textfile. On return from \$B080, the low byte of the variable location is in the accumulator, and the high byte is in the Y-Register. We store this in zero page \$40/44, which is the system variable pointer (line 2118-2120).

Next we get the line number and store it in zero page \$62/63, which is the floating point accumulator #1 (line 2160-2161). The next routine (\$B399) which we call converts the line number into a proper floating point number (line 2250). We have to do it in this rather roundabout way because variables of this type have to be stored as floating point numbers in order to be recognized later on!

Finally we call \$B8D0 (line 2260), which stores the value contained in flip acc #1 in the variable area itself. Now our label has been stored like any other Basic variable in the variable store, including the number of the line on which it appeared. We can return to our main search loop and look for the next label.

The fact that each of the labels we

declared is stored like any other Basic variable means, of course, that only the first two characters of the label will be recognized. The rest of the label is ignored, which limits the use of labels

rather, because it doesn't give much scope for declaring meaningful names which are recognized.

Therefore, in the next article I'd like to develop a routine which builds

and recognizes a symbol table map from the Basic variable area. This will enable us to declare labels with six or more characters which are fully recognized.

Listing 1

```

10      ORG 40000
20      END
30
40      CHARGET EQU 40073
50      EXECUTET EQU 40090
60
70
80
90      TURN EXTENDED BASIC ON
100     BY CHANGING VECTOR AT 40000
110
120      EXTRASON LDA #PRGSTART
130      STA <EXECUTET
140      LDA #PRGSTART
150      STA <EXECUTET
160
170
180      RTS
190
200
210      TURN EXTENDED BASIC OFF
220     BY CHANGING VECTOR AT 40000
230     BACK TO NORMAL [40074]
240
250      EXTRASOFF LDA #B4074
260      STA <EXECUTET
270      LDA #B4074
280      STA <EXECUTET
290
300
310      RTS
320
330
340
350     *** MAIN PROGRAM ENTRY ***
360
370     LOOK FOR EXTENDED BASIC COMMANDS
380
390     PRGSTART JIR CHARGET
400     JIR EXECUTET
410     JIF B4074
420
430
440
450     EXECUTET CMP #0
460     BEI NEXT
470     JIF OFF RT
480     NEXT CMP #4000 'GOTO'
490     BEI GOTO RT
500     CMP #4004 'RUN'
510     BEI NEXT1
520     CMP #4008 RT
530     NEXT1 CMP #4000 'GOSU'
540     BEI GOSU RT
550     CMP #
560     BEI SPEC RT
570
580
590
600     CMP #0
610     BEI NORMAL
620     JIR CHARGET
630     CMP #0
640     BEI NORMAL
650     JIR CHARGET
660     CMP #L

```

```

670     BEI NORMAL
680     JIR CHARGET
690     CMP #4000 'COS' TONES
700     BEI COLOR RT
710
720
730
740     DO NORMAL NON-ROUTINE
750
760     NORMAL JMP B4074
770
780
790     EXECUTE 'COLOR' COMMAND
800
810
820     SET INK PARAMETER
830
840     COLOR RT JIR CHARGET
850     JIR B4074
860     JIR B4077
870
880     CHANGE INK COLOUR
890
900
910     STY 444
920
930     SET PAPER PARAMETER
940
950
960     JIR CHARGET
970     JIR B4074
980     JIR B4077
990
1000    CHANGE PAPER COLOUR
1010
1020
1030     STY 43201
1040
1050     SET BORDER PARAMETER
1060
1070
1080     JIR CHARGET
1090     JIR B4074
1100     JIR B4077
1110
1120    CHANGE BORDER COLOUR
1130
1140
1150     STY 43200
1160
1170    JUMP TO REST OF NON-ROUTINE
1180
1190
1200     RTS
1210
1220
1230     EXECUTE 'GOSU' COMMAND
1240
1250     GOTO RT JIR CHARGET
1260     GOTO RT1 JIR B4074
1270     JIR B4077
1280     JIR B4074
1290     RTS
1300
1310     EXECUTE 'GOSU' COMMAND
1320
1330     GOSU RT JIR CHARGET
1340     LDA #4000
1350     JIR B4074
1360     LDA #0
1370     PLA
1380     PLA
1390

```

0.0 1.0 2.0 3.0 4.0 5.0 6.0 7.0 8.0 9.0 10.0 11.0 12.0 13.0 14.0 15.0 16.0 17.0 18.0 19.0 20.0 21.0 22.0 23.0 24.0 25.0 26.0 27.0 28.0 29.0 30.0 31.0 32.0 33.0 34.0 35.0 36.0 37.0 38.0 39.0 40.0 41.0 42.0 43.0 44.0 45.0 46.0 47.0 48.0 49.0 50.0 51.0 52.0 53.0 54.0 55.0 56.0 57.0 58.0 59.0 60.0 61.0 62.0 63.0 64.0 65.0 66.0 67.0 68.0 69.0 70.0 71.0 72.0 73.0 74.0 75.0 76.0 77.0 78.0 79.0 80.0 81.0 82.0 83.0 84.0 85.0 86.0 87.0 88.0 89.0 90.0 91.0 92.0 93.0 94.0 95.0 96.0 97.0 98.0 99.0 100.0

Listing 2

```

115 BEGINNING
120 FOR INT=1 TO 10
130 PRINT INT; "PAGES: "
140 TAB(12); PRINT INT; "PAGES: "
150 PRINT TAB(15); "PAGES: "
160 FOR INT=1 TO 10
170 PRINT TAB(18); "PAGES: "
180 GOTO 190
190 GOTO 200
200 GOTO BEGINNING
210 .DELAY
220 FOR N=1 TO 500
230 NEXT
240 RETURN

```

Tech Troubles

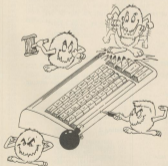
*A selection of the problems solved and readers' hints
from this month's mailbag*
By Andy Andros

Why do computers use integers as well as normal decimal values, and where do floating point numbers come into it?
David Kennedy, Easingwold

Dear David,
Another term for an integer is a 'whole number' or, as you point out, a number without any decimal point. Other numbers are referred to as 'floating point' numbers because of the way in which the mathematics is performed in the computer's memory. Floating point calculations are difficult to explain in the space allowed here but, luckily, the fact that they do work is all that you really need to know for Basic.

The reason that computers use integers is because it helps to speed-up arithmetic, and is more accurate than floating points where a certain amount of rounding up and down of values occurs. The speed benefit is gained because there is no need to calculate the position of the decimal place, or to check if the value should be expressed in exponential form, when an extremely large or small number is expressed by a value, and the power to which it has to be raised to reveal the actual number.

A popular myth is that integers use less memory than floating point values, but this is not entirely true. Integers use four bytes - two for the variable name, and two for the actual number. Floating point numbers similarly use two bytes for the variable name, but five more bytes are needed for the value, so, on the face of it,



integers do need less space, but the integer value is followed by three unused bytes, which means that both types of variable physically occupy seven bytes of memory.

I thought the Your Commodore Serious Users Guide 1988, and found the 64 tip for the 64 quite informative. One of the tips is how to simulate a PRINT AT command without using rest of cursor down and left. I thought that you may be interested in a technique I've been using for the past few years:

```
10 POKE14,1:PRINT"HELLO"
20 POKE10,15
30 PRINT"HELLO"
```

Location 14 is the vertical position of the cursor on the screen (value 1 to 24). The PRINT command in line 10 is used to update this new row in the Commodore's memory, and a cursor up to replace the cursor onto the required line. Next the cursor is placed horizontally by poking a value in location 20 and the message can then be printed.

Two final words on this routine refer to printing on the top and bottom lines. For using the top line, pointing a cursor home with a semicolon will set the row position and then the column value can be poked into 20. If the last line is used the screen will scroll up unless you follow the message printing command with a semicolon.

I hope this is of use to someone, somewhere.

Carol Seddon, Wigan

Dear Carol,

Thanks very much for that tip. I found it very informative, and if anyone else has a technical tip I'd be very pleased to hear about it.

In a listing I came across the following system which I don't understand:

```
100 INDEX=CHR$(1)+48-1:GO TO 470
```

Could you explain what it achieves?

Clive Fowler, Withers

Dear Clive,

This is an unusual but useful application of a comparison command. It is the sort of routine which would convert a decimal value to its hexadecimal equivalent.

To understand how it works, I'll have to explain the principle behind the greater than and less than commands. When a comparison is made, the result is a zero if the condition is false and a minus one if the condition is true.

In the example that you've suggested, the value given by the comparison would be zero if X had a value between zero and nine but minus one if the value exceeds nine. What the equation is doing is taking the value in A and adding 48 to give a value whose CHR\$ equivalent is the number character which corresponds to X. This works well up to a value of nine, but values 10 to 15 would produce punctuation marks and symbols because there is a gap of seven characters between the CHR\$ value for nine and the CHR\$ value for the letter A. The program has to be able to add seven to values over 10, but not to values below that.

The solution is to compare the value with nine, and then multiply this by seven. This would mean that the low values would subtract two from the CHR\$ value but high values would subtract -7 (i.e. multiplied by 7). Subtracting a negative value follows the mathematical law that states that two minuses gives a plus, so the program actually adds seven to the CHR\$ value.

To test this try setting up a loop for A having values of zero to 15. Place the equation inside the loop and a statement to print each INDEX value, and the result will be a list of hex equivalents from 0 to F.

I have written a program which uses several groups of DATA statements but the only problem with it is that RESTORE can't be set to point to a particular line. Is there any way around this, because at the moment I have to RESTORE and then read

in DATA until the correct point is reached?

Harry Ivy, Hampton

Dear Harry,

There are two ways to tackle your problem. The first is to read the data into several arrays and manipulate these instead of using data READs.

The second method is to create a special form of the RESTORE command. This is done by loading in the data until the line where the program is to be restored to. The actual memory location can be stored by poking locations 65 and 66 and storing this in a table in a safe part of memory.

Each suitable point is similarly stored until the table is complete. When the data reading has to be restored to any of these points, they can be poked from their storage point and poked back to 65 and 66.

Can you tell me where I can find a good book containing a breakdown of the C128 Basic ROMs? I have a hunch that such a book may not exist, because I have a book which gives a breakdown of the Kernel, but which states that the Basic disassembly would fill a very large volume.

Richard Terry, York

Dear Richard,

You'll be pleased to hear that such a book does actually exist, though you are right in assuming that it is a weighty tome. C128 Basic 7.8 Breakdown, published by Abacus, runs to over 680 pages. Of this, 450 pages contain an annotated ROM disassembly, and the rest is packed with useful information about the workings of the system, alongside some useful programming hints and routines. The price is £19.95, and it is available from Precision Software, 6 Park Terrace, Worcester Park, Surrey KT4 1JZ, but please include £1.49 for post and packing.

If you have a problem let us know and Andy will try to help. Write to: Tech Trouble, Your Commodore, Argus House, Boundary Way, Hemel Hempstead, Herts HP1 1ST.

THE EPSON SQ-2500

The first major task is getting the SQ-2500 out of the box - ideally, it's a job for two people. The SQ-2500 might be expensive, but it has a wide carriage and you certainly get a lot of printer for your money. Styling is recognizably Epson, but the greatest impression from the first is that this printer means quality. Construction is superb. I didn't actually try it, but I'm sure I could jump on the SQ-2500 without damaging it (and I weigh over 16 stones).

Setting it up

This is an ink-jet printer so there's no ribbon, just an ink cartridge, paper feed knobs, the mains lead and paper guide. Open a cover on the right side towards the rear of the printer, push in the ink cartridge, close the cover and that's done. On the left side is a similar cover, but unless you want to use either of the possible two extra font cartridges, you needn't open it.

The interface and power lead connect at the rear, as usual for Epson, and the power switch is on the right side. The paper guide easily slots into position, and when an new ink cartridge is fitted, the SQ needs priming (this is the long cleaning cycle mentioned later, but it just means switch on and press a button). After about half a minute it's ready to go.

Both parallel and a serial interface are standard, with provision for a third one, and any of the entire Epson interfaces (listed in the LQ-580 report) can be used too. The standard buffer is 8K. The manual is 178 pages, in contrast to the LQ's 238, but contains all the necessary ingredients, including both types of command summary and a quick reference card.

If you can afford an SQ, buy one! If you do, don't waste time looking for disadvantages - there aren't any! Four buttons (and one of these is an off line control) EVERYTHING! This explains the short manual. There's very little on trouble-shooting.

To conclude last month's feature on the world of 24 pin dot matrix printers.

Robin Burton admires the highly impressive Epson SQ-2500

The only way I can envisage anyone having trouble with the SQ is if they can't read, in which case the manual wouldn't help much.

I'm not being flippant! It's all set-up, and manual selection is by question and answer, but with a difference. There's a liquid crystal display at the front of the printer used for the selection buttons, and the SQ asks the questions. You answer by simply pressing a button, and the settings are stored. This is a dream machine.

Superb though this is, there's more! The SQ also has four macros. Each of these is a complete, permanently stored definition containing everything you could want the printer to know about a job. Any one of the four can be loaded automatically at power-up; you choose which one you want in the configuration defaults, which remain permanently set unless you alter them.

You can manually load any macro by a couple of button presses, and can also amend them manually, by software or both, either temporarily, or permanently. Simply re-save a macro at any time (by pressing a button) if you want the changes to be permanent. If not, they're forgotten, either when you switch off or when you load a different macro. You might also want to print out the settings, in which case - press a button.

This is all so comprehensive yet simple that describing it is difficult. I've therefore included a print of my configuration settings separately, so you can see for yourself. Everything is stored - left and right margins, font, style, pitch, page size, etc, etc.

As standard, the SQ-2500 is a cut sheet machine, but an optional tractor unit can be added for continuous paper. You may have gathered by now that the SQ-2500 is intended for high quality, high speed, high volume output. The fact that the tractor is top-mounted and therefore without paper parking is, in context, irrelevant.

Don't misunderstand - swapping between continuous and sheet-feed is easy enough, but if you need to do it very often you don't need an SQ-2500. It wasn't built for the average home user's mixed low-volume needs, and used that way would be like doing the grocery shopping in a Formula One car, theoretically possible but...

If cut sheet is the major use a double-bin automatic sheet feeder can also be attached.

Specification

The SQ-2500 is of course above all an Epson, so the compatibility comments for the LQ-580 all apply to the SQ, including the 12 national character sets.

Six fonts are standard for the SQ-2500, such as usual with Epson. Additional effects are limited to double width, double height or both. All fonts are available in the usual 10, 12 and 13 cpi, and all can be condensed except 13 pitch.

There's also provision for two font modules, but so far as I could see all the Epson LQ fonts are standard in the SQ except OCR-B. Minimum vertical spacing is 180ths of an inch, and horizontal is 360ths.

Using the Epson

When the SQ powers-up it automatically goes through a self-cleaning cycle. The printer informs you (via the LCD) when the ink cartridge is getting low, and when it's exhausted. According to the manual, a cartridge lasts for 3,000,000 characters in LQ, and 6,000,000 in draft. This seems fair enough - I've gone through well over 1,500 sheets, largely in LQ, (roughly 6,000,000 characters) and not a word so far.

Using semi-automatic sheet feed is simple and quick, and it has been absolutely reliable with all weights of paper. Just drop a sheet into the guide, press 'form feed' and the sheet is lined up perfectly (I even tried it with a 9 x 4 inch envelope - notoriously difficult to keep straight - with no trouble at all).

If the tractor is fitted, it simply attaches on top of the SQ - there are no covers to change. Usefully, along with the tractor, Epson provide a matching base on which the printer sits. Continuous paper is kept inside this, out of the way. A paper rest is also included to keep the paper clear of the heads.

Manual line feed, form feed, self-test and hex dump are controlled by the buttons, and the self test includes the current configuration and macro settings too. As mentioned before, just the configuration and the four macros can also be listed on demand (or alternatively checked directly via the LCD). Without altering any of the current settings you also can switch between draft or LQ at any time by pressing either the line-feed or the form-feed button.

Print quality is frankly so superior to that of an impact dot-matrix printer that a comparison is pointless. Because there are no wires passing through a fabric ribbon, the individual dots are much smaller and more precise. I've

even asked when I bought a laser-printer by people who didn't know I had the SQ. It might not be quite that good, but it's obviously near enough, and you'd have to put the two side by side to notice a difference. Also unlike any impact printer, the quality never varies. There's no deterioration or fading of print quality, because there's no ribbon to wear out. Characters are always perfect, and uniformly sooty black.

The SQ is the fastest matrix printer I've ever seen. The figures don't adequately tell the story - you have to see it to appreciate just how quick it is. In fact its speed was at one point an inconvenience. With such dense character images, double strike is almost unnecessary. Of course I tried it, but output was quick enough to allow the paper to re-fold in the collection basket before the ink had fully dried. I have now lowered the basket.

The final difference about the SQ is noise, specifically the absence of it - only ink touches the paper. You don't realise just how much you are mentally conditioned to brace yourself for the onslaught of noise from an impact printer, at least not until you start printing and it doesn't happen. It's rather uncanny at first, and after years of impact printers, it took me a week to get used to this. Needless to say the SQ easily passes the telephone test, in fact if you were surrounded by them it would be no problem.

Quite aside from its obvious desirability, this aspect of the SQ's performance (along with the others) offers very serious benefits for offices where several printers are at work. I would think the amount of noise from the SQ (just the carriage moving back and forth) would be acceptable even in places like libraries and hospitals too.

The SQ-2500 has operated perfectly throughout.

Conclusion

It hardly needs saying that the SQ-2500 is Epson's top-of-the-range conventional dot matrix printer, and it has been for a few years now. It is built like the Fortk Bridge, and has by far the easiest manual control of all operational features of any printer I know. Macros are convenient, and allow an automated switch between four completely different setup definitions in literally a couple of seconds.

Of course the recent arrival of low-cost laser printers must have ended SQ sales, but there are still plenty of jobs that laser printers don't do very well or at all, (like printing A3 pages sideways), and their running costs are also higher than the SQ's. One laser printer limitation however is shared by the SQ-2500. Because it's not an impact device, multi-part sets are impossible, though with its speed you'd probably just print extra copies as needed.

In its target market, I doubt that the SQ-2500 has any competition. Quality is beyond question in construction, operation and output. It must be the fastest, quietest way of getting high quality print onto paper, with all the flexibility of a conventional printer and a virtual absence of operator skill or training.

The recommended retail price is £1,349. Options include a tractor unit at £90 and a double-bin automatic sheet feeder at £170. (£1,519, £5, £103.50 and £425.50 including VAT.) Ink cartridges are about £24, and have a shelf life of two years.

Checking current advertisements I found that the SQ-2500 can be purchased for around £975 plus VAT, with proportional reductions in the optional fittings.

TABULATION FOR YOUR COMMODORE/MAT/PRINTERS:

	Dimensions WxDxH	Time/Speed to print 5000 chars.		Weight
		Draft	Letter	
Star LC14-00	17"x11"x4.5"	54 sec/58 cps	173sec/36 cps	34.5 Lbs
Epson SQ-500	16.5"x12"x6"	52 sec/54 cps	125sec/40 cps	15.5 Lbs
Citrus HQPrint	24"x3"x4.75"	45 sec/111 cps	180 sec/30 cps	36.5 Lbs
Epson SQ-2500	23.5x15x8	32 sec/136 cps	56 sec/89 cps	25 Lbs

Sketchpad 128



*Gordon Davis examines
a rare specimen – an
80-column graphics
package for C128 owners*

Skechpad 128 is a rare animal. Almost all Graphics packages for the C1280 operate in 80-column high-resolution mode, but this one is different – it offers a full graphics capability in 80-column mode – a resolution of 640 pixels horizontally.

The package has been made possible by the release of Walmusch's Basic 3, a language package which we reviewed last year. Basic 3 is designed as a language which offers a graphics-orientated environment to the system designer. It's no surprise, therefore, that the salient features of Sketchpad are exactly those which Basic 3 supports most readily.

A Commodore 128 mouse is a necessity for Sketchpad. This, I think,

is unfortunate, since it isn't the sort of item that Commodore owners are likely to just have hanging around. Although, as the authors point out, a mouse is by far the best device for doing graphics, it would have been kinder to allow a joystick to be interfaced with the program, just to get people started.

Little messing

Fortunately, we managed to find the office's 1251 mouse, so that play could begin. There's very little messing around following loading. Sketchpad moves straight into the drawing screen following the obligatory Basic 3 loading screen.

Free Spirit, which, as you might guess, is a Californian software house, has obviously done its Apple Macintosh homework. The main screen looks very Mac-like. All menu selections are made from the drawing screen by clicking the left mouse button. The right button is reasonably consistent in returning the user to the main menu.

The main elements on the main menu are exactly those which have been made familiar by that universal program, MacPaint. Naturally, you can draw firsthand on the screen with a variety of different pens. The indispensable airbrush is also there. The annoying thing about this is that the airbrush area cannot be sized, which is something that you can see even with C64 packages.

Sketchpad supports a suitably fast fill. This is a little worrying in the way it operates, since it seems to leave many lines unfilled along the way. But never fear, the filling algorithm gets back and fills them all back in too.

A multi-font option for writing is also available. Different fonts can be loaded from disk, and can be typed anywhere on the screen in one of five different sizes. A useful touch is that Return will align the text cursor exactly one line below the text you've already put in.

This is not a multi-colour graphics program, quite rightly, because unless you are designing loading screens or demos, anything other than two-colour screens is functionally useless

for all owners of black-and-white printers. Even so, you can also alter the foreground/background to give you more screen readability.

Although clearing the screen is fairly doddle on Steepcad - there is no clock as this, you can lock your program into memory, which will put it into a spare area of Ram for safety. This is also useful if you're uncertain about the changes you're making to a picture.

The graphics shape options are the ones that show the most difference over most graphics packages. All the traditional favourites are there: lines, boxes, circles, and so on. Freshening these is more or less as straightforward as the commands on any other graphics package. A unique extra, however, is the addition of the so-called Rylander shapes.

Solid graphics

Rylander shapes are views of shaded solids of three types: cylinders, spheres and torus. They are named after the programmer who designed a set of

algorithms for representing them easily.

Basic 8 later incorporated these calculations, and here they are at last in a graphics package. The shading on the Rylander shapes can be varied between pixel dots or smooth shading, and whatever you use them for, they are an interesting addition.

Besides these shapes, another useful feature is the inclusion of axes. You can provide axes of any pitch and angle. Rays can also be provided, subordinated at any angle from a central point. Both of these make the production of piecharts an easy matter.

One vital thing I spent ages looking for was some form of Zoom option. It's very unpleasant using a graphics package, and some means of adjusting things on a pixel-by-pixel basis can be useful. This is, annoyingly, not available from the main screen, but from a sub-menu called the Gadgets menu.

Besides a pixel menu, Gadgets also supports a Cut/Paste option which allows you to move chunks of the picture around, or simply remove them. You can also scroll your picture - an interesting feature which few

packages support. This menu also supports a number of patterns which can be used to fill areas. Unfortunately, you have to draw the clip patterns in yourself - there is no automatic fill for these.

Free Spirit deserves commendation for the number of features which have been included in Steepcad - I haven't covered everything here. The main flaw in the program is that it is a trifle slow, due to being written entirely in Basic 8.

I'm also not sure about some of the design priorities - some of the commands are very hard to get to. The program documentation is brief and unhelpful. These days, this is something I expect software houses to get right. On the other hand, C128 88-column drawing packages are fairly hard to come by, so it seems unfair to criticise too much.

Footnote:

Title: Steepcad 128. Supplier: FEEL/Free Spirit, 18 High St., Poole, Dorset, Weymouth, DT99 1JG. Tel: (0594) 334133. Price: £24.95.

DON'T GET IN A TANGLE..

take out a subscription

Make sure of your regular copy of *Micro Music*, with all the news, views and reviews from the computer music scene, delivered to your door at no extra cost in the UK. Just fill in the form, enclose your remittance and you're in touch.

4 issues	UK	£10.50
4 issues	Europe	£14.00
4 issues	Middle East	£14.15
4 issues	Far East	£15.50
4 issues	Rest of the World	£16.00
4 issues	USA	\$25.00

Air Mail rates given on request. Overseas subscription includes postage.



Please send my subscription to: *Micro Music* with the London air despatch order to: *Micro Music* with the

NAME (PRINTED) ADDRESS

Signature

Send your remittance to: 2500597, 5 River Park, Essex, CM9 7PL, 0774 1000007



Routine Programming

*A bubbly routine
to sort lists into
orderly sequences*

By Eric Doyle

Programs often include lists of numbers or strings which have to be displayed in an orderly way. This could be alphabetically or numerically. This bubble sort subroutine can be used, suitably modified, for either purpose.

Bubble sorts work by comparing neighbouring list items and swapping them over if one exceeds the other. Take this list as an example:



The first number is compared to the rest of the numbers in the list one at a time. If the number under comparison is smaller than the first number, a straight swap takes place. In this way the smallest value rises to the top.



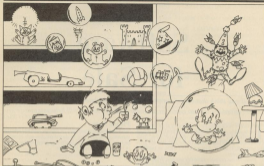
After this process is completed, the second number is compared to all of the following figures in a similar way.



This is repeated, gradually passing down the column until the last two characters are compared and swapped if necessary, and the sort is then complete.



The two elements of the sort are a loop which selects the character to be tested, and a loop within this, a nested loop, which selects the remaining characters in the list for comparison.



```
60000 FOR%1=1TOZ-1
60010 Z%-Z1+1
60020 FOR%2=1TOZ
```

The value Z is the number of items in the list. Line 60000 forms the main loop for the characters to be tested. It only runs from the first to the master penultimate character because there would be no point in comparing the final character with itself.

Line 60010 calculates the first character to be tested by the nested loop which is initiated in line 60020. Line 60020 tests the two values, and if they are already in the correct order no action is taken and the exchange routine is bypassed.

```
60030 IF(Z1<Z2)THENSWAP(Z1,Z2)
```

If an exchange is necessary, this is done by using a temporary store for one of the values, so that values can be swapped with ease.

```
60040 Z3%=Z1%
60050 Z1%=Z2%
60060 Z2%=Z3%
```

Z3% takes the value held in Z1% and Z2% can then be given the value held in Z3%. The stored value can then be moved from Z3% to Z2%, and Z3% may then be discarded.

Now that everything has been done, the loops can be closed and

eventually control is handed back to the program with a RETURN command.

```
60070 NEXTZ,Z1
60080 RETURN
```

The listings shown contain an example of numerical sorting using this sub-routine and there is also an alphabetical sort routine which shows how this routine has to be modified for this use. Really all that has happened is that the Z(s) values are changed to Z\$(s).

Parameters for Main Program

Sendings

- Z Number of list items
- Z(s) List item values

Returns

- Z(s) The sorted list

Other Variables Used

- Z1 Main loop variable
- Z2 Remainder of list items
- Z3 Nested loop variable

PROGRAM: NUMBER SORT	
01 10 0-1:40-50-1:100000,221000	20 70 FOR%1=1TOZ:PRINT(Z1%);NEXT
02 20 0-1:40-50-1:100000,221000	30 80 END
03 30 0-1:40-50-1:100000,221000	40 1000 FOR%1=1TOZ-1
04 40 0-1:40-50-1:100000,221000	50 1010 Z%-Z1+1
05 50 0-1:40-50-1:100000,221000	60 1020 FOR%2=1TOZ
06 60 0-1:40-50-1:100000,221000	70 1030 IF(Z1%<Z2%)THENSWAP(Z1,Z2)
07 70 0-1:40-50-1:100000,221000	80 1040 Z3%=Z1%
08 80 0-1:40-50-1:100000,221000	90 1050 Z1%=Z2%
09 90 0-1:40-50-1:100000,221000	100 1060 Z2%=Z3%
10 100 0-1:40-50-1:100000,221000	110 1070 NEXT%2,Z1
11 110 0-1:40-50-1:100000,221000	120 1080 NEXT%1
12 120 0-1:40-50-1:100000,221000	130 1090 RETURN

PROGRAM: NAME SORT	
01 10 0-1:40-50-1:100000,221000	20 80 END
02 20 0-1:40-50-1:100000,221000	30 1000 FOR%1=1TOZ-1
03 30 0-1:40-50-1:100000,221000	40 1010 Z%-Z1+1
04 40 0-1:40-50-1:100000,221000	50 1020 FOR%2=1TOZ
05 50 0-1:40-50-1:100000,221000	60 1030 IF(Z1%<Z2%)THENSWAP(Z1,Z2)
06 60 0-1:40-50-1:100000,221000	70 1040 Z3%=Z1%
07 70 0-1:40-50-1:100000,221000	80 1050 Z1%=Z2%
08 80 0-1:40-50-1:100000,221000	90 1060 Z2%=Z3%
09 90 0-1:40-50-1:100000,221000	100 1070 NEXT%2,Z1
10 100 0-1:40-50-1:100000,221000	110 1080 NEXT%1
11 110 0-1:40-50-1:100000,221000	120 1090 RETURN

Software for Sale

If you think that one of our programs looks very interesting, but you can't afford the time to type it in, then our software service will help you out

It's three o'clock in the morning. You sit at the computer keyboard having just finished a marathon typing session. Entering one of the superb programs from *Four Commodore*. Your fingers reach for the keyboard and press the letters R, U and N. You press RETURN, sit back and nothing happens.

Everyone has probably faced this problem. When it does happen it's a matter of spending hours searching through the program for any typing mistakes. No matter how long you look or how many people help you, you can usually guarantee that at least one little bug slips through unnoticed.

The *Four Commodore Software Service* makes available all of the programs from each issue on both cassette and disk at a price of £6.00 for disk and £4.00 for cassette. None of the documentation for the programs is supplied with the software since it is all available in the relevant magazine. Should you not have the magazine then back issues are available from the following address:

INFUNET LTD, 3 River Park Estate, Redditch, Werts,
HP4 1HL
Tel: (04427) 7664

Please contact this address for prices and availability.

The Disk

Programs on the disk will also be supplied as totally working versions, i.e. when possible we will not use Basic Loaders thus making one of the programs much easier. Unfortunately at the moment we cannot duplicate C16 and Plus/4 cassettes. However programs for these machines will be available on the disk.

What programs are available?

At the top of each article you will find a strap containing the article type, C64 Programs etc. So that you can see which programs are available on which format, you will also find a couple of symbols after this strap. The symbols have the following meaning:



This symbol means that the program is available on cassette.



These programs are available on disk.

Please Note

Since the programs supplied on cassette are total working versions of the programs, we do not put disk-only programs on tape. There is no sense in placing a program that expects to be loading from disk on to tape.

JANUARY 1989

PREPARE SPRITES - A powerful sprite editor for the C64.

HAZPRO - A simple but helpful text processor for the C64. Available on disk and cassette but will only move files on tape.

UDG COMPRESSOR - Save on memory when using UDG's in your programs. For C64 only.

WILLIAM TELL - Our popular arcade game for the C64.
+4 AUTOLOAD - Improve tape loading on your Plus/4 cassette. Only available on disk.

MINIBASE - A database for C128 owners.

ORDER CODE

DISK YJAN89 £6.00

TAPE YJAN89 £4.00

FEBRUARY 1989

TAPE MENU - Add a menu system to your program cassettes (C64).

SOUND EFFECTS - A superb sound editor for the C64.

F DUMP - Dump your C64 text screens to printer with ease.

DATA LOADER - A simple way to enter those screens of C64 DATA files.

SPELLE LIBRARY - A collection of birds to your growing library (C64).

PLAY THE GAME - A superb fruit machine program for the Plus/4. (Available on disk only).

ORDER CODE

DISK YFEB89 £6.00

TAPE YFEB89 £4.00

MARCH 1989

PERSONAL FILE - A cross between a wordprocessor and a database that allows you to set up "cards" that can be quickly altered (C64 Disk only).

LETTER WRITER - An 80 columns word editor for writing those personal letters (C64).

BASIC WORKSHOP - A single key entry system, just like a Spry (C64).

HEAD FOR HOME - Our version of a popular board game for C16 and Plus/4 users - available on disk only.

SPRITE LIBRARY - Geometric shapes from this month's installment (C64).

ELECTRONIC NOTEBOOK - A personal diary on disk (C64 disk only).

WILLIAM TELL - Our very popular arcade adventure for the C64.

ORDER CODE

DISK YDMAR89 £6.00

TAPE YCMAR89 £4.00

APRIL 1989

BASEX - Give your C64 new sound, graphics and touch commands as well as a machine code assembler with this Basic extension.

AUTOSCROLL - Professional text screen scrolling with this C64 utility.

BALANCE SHEET - Keep your bank manager happy by keeping better track of your money with this C64 program (disk only).

HON 64 - Add icon control to your own programs with this C64 utility.

88 COLUMNS - A suite of 88-on graphic commands for C128 80 column users.

REUS - A disk file copier for Plus/4 users. Available on disk only.

MULTI-PRECISION ARITHMETIC - A suite of powerful mathematical routines expanding your C64's number crunching abilities.

STORAGE SPACE - Use the RAM behind the C64's Kernal and Basic ROMs for data storage (available on disk only).

ORDER CODE

DISK YDAPR89 £6.00

TAPE YCAPR89 £4.00

MAY 1989

ANTI-FREEZE - Protect your C64 programs from cartridge-based loaders.

MULTI-COLOURED LIST - Brighten up your C64 Basic listings by adding on-screen colour.

PRINTFX - A suite of powerful screen printing commands to add to your Plus/4 Basic (disk only).

BANKER 128 - Money Management for C128 disk users.

TU HRO TOUCH - A superb typing trainer for C64 disk users.

ORDER CODE

DISK YDMAY89 £6.00

TAPE YCMAY89 £4.00

Coverages on disks are available from March 1986. Please ring the editorial office (01-477 0626) for details of these.

ORDER FORM - PLEASE COMPLETE IN BLOCK CAPITALS

NAME	QTY	TAPE/DISK	ORDER CODE	PRICE
JUNE '89		TAPE (£4.00)	YDJUNE '89	
JUNE '89		DISK (£6.00)	YDJUNE'89	
OVERSEAS POST £1				
			TOTAL	

NAME.....

ADDRESS.....

POSTCODE.....

I enclose a cheque/postal order for £..... made payable to ARGUS SPECIALIST

PUBLICATIONS LTD.

All orders should be sent to: FOGW COMMODORE

READERS SERVICES, ARGUS HOUSE, BOUNDARY WAY, HEMEL HEMPSTEAD, HERTS. HP1 1ST

Please allow 28 days for delivery.

YOUR COMMODORE

Lineage: 58p per word.

(including VAT)

Semi display: £10.95 plus VAT per single column continuous minimum 3cm. Ring for information on series bookings/discounts.

An advertisement in this section must be prepaid.

Advertisements are accepted subject to the terms and conditions printed on the advertisement rate card (available on request).



0442 66650

Send your requirements to:
CLASSIFIED DEPARTMENT
ASPLED, ARDUGH HOUSE,
BOUNDARY WAY,
HEMEL HEMPSTEAD HP2 7ST.

SOFTWARE



**BARNHALL
COMPUTERS LTD**

Unit 10, The Old
Mill, Barnhall, Wokingham, RG40 3JF

0442 666700 (Fax 0442 666701)

THE BEST SOFTWARE FOR

COMMODORE PLUS/SEAMUS

Ring now for a free quote on our 2021

Table 1.

To obtain this information and prices and

more information

contact Barnhall Computers Ltd

at Aspley Way, Hemel Hempstead, Herts, SG9 7ST

SERVICES

COMMODORE SUPPLIES

500 Power Supply	£19.95
5000 Plus + Power Supply	£29.95
10000 Plus + Power Supply	£39.95
10000 Plus + Power Supply	£39.95
10000 Plus + Power Supply	£39.95
10000 Plus + Power Supply	£39.95
10000 Plus + Power Supply	£39.95
10000 Plus + Power Supply	£39.95
10000 Plus + Power Supply	£39.95
10000 Plus + Power Supply	£39.95

Prices include VAT + P&H allow up

to 14 days for delivery. Cheque or

credit card only. Please allow 10 days

for delivery. Tel: 0442 666700.

HARDWARE

COMMODORE COMPUTERS FOR REPAIRS

Call today for the best prices on the

area.

10000 Plus + Power Supply

10000 Plus + Power Supply

10000 Plus + Power Supply

10000 Plus + Power Supply

10000 Plus + Power Supply

10000 Plus + Power Supply

10000 Plus + Power Supply

10000 Plus + Power Supply

SOFTWARE

YOUR COMMODORE SPECIALS TECH DRAW 64 - A

comprehensive technical

illustration aid for 64

owners for both disk and

tape. Commands available

include LINE DRAW, RAYS,

BOX, CIRCLE, ARC,

ELLIPSE, HYPERBOLA, FIL,

HATCH & TEXT. With copy,

print and paste, save and

load or print your drawing at

the 64-66 tape or disk. Order

Code: TDR64 (8pp) TDR64

(18p)

Full details in Your

Commodore July 1989.

SPECIAL OFFER

T.C. COMPUTERS

1st Wharfedale 10000 Plus

10000 Plus + Power Supply

10000 Plus + Power Supply

10000 Plus + Power Supply

10000 Plus + Power Supply

10000 Plus + Power Supply

10000 Plus + Power Supply

10000 Plus + Power Supply

10000 Plus + Power Supply

10000 Plus + Power Supply

10000 Plus + Power Supply

10000 Plus + Power Supply

10000 Plus + Power Supply

10000 Plus + Power Supply

10000 Plus + Power Supply

10000 Plus + Power Supply

10000 Plus + Power Supply

10000 Plus + Power Supply

10000 Plus + Power Supply

10000 Plus + Power Supply

10000 Plus + Power Supply

10000 Plus + Power Supply

10000 Plus + Power Supply

10000 Plus + Power Supply

10000 Plus + Power Supply

10000 Plus + Power Supply

10000 Plus + Power Supply

10000 Plus + Power Supply

10000 Plus + Power Supply

10000 Plus + Power Supply

10000 Plus + Power Supply

10000 Plus + Power Supply

10000 Plus + Power Supply

10000 Plus + Power Supply

10000 Plus + Power Supply

10000 Plus + Power Supply

10000 Plus + Power Supply

10000 Plus + Power Supply

10000 Plus + Power Supply

10000 Plus + Power Supply

10000 Plus + Power Supply

10000 Plus + Power Supply

10000 Plus + Power Supply

10000 Plus + Power Supply

10000 Plus + Power Supply

10000 Plus + Power Supply

You'll never believe OUR PRICES

10000 Plus + Power Supply	£19.95
10000 Plus + Power Supply	£19.95
10000 Plus + Power Supply	£19.95
10000 Plus + Power Supply	£19.95
10000 Plus + Power Supply	£19.95
10000 Plus + Power Supply	£19.95
10000 Plus + Power Supply	£19.95
10000 Plus + Power Supply	£19.95
10000 Plus + Power Supply	£19.95
10000 Plus + Power Supply	£19.95

10000 Plus + Power Supply

10000 Plus + Power Supply

10000 Plus + Power Supply

10000 Plus + Power Supply

10000 Plus + Power Supply

10000 Plus + Power Supply

10000 Plus + Power Supply

10000 Plus + Power Supply

10000 Plus + Power Supply

10000 Plus + Power Supply

10000 Plus + Power Supply

10000 Plus + Power Supply

10000 Plus + Power Supply

10000 Plus + Power Supply

10000 Plus + Power Supply

10000 Plus + Power Supply

10000 Plus + Power Supply

10000 Plus + Power Supply

10000 Plus + Power Supply

10000 Plus + Power Supply

10000 Plus + Power Supply

10000 Plus + Power Supply

10000 Plus + Power Supply

10000 Plus + Power Supply

10000 Plus + Power Supply

10000 Plus + Power Supply

10000 Plus + Power Supply

10000 Plus + Power Supply

10000 Plus + Power Supply

10000 Plus + Power Supply

10000 Plus + Power Supply

10000 Plus + Power Supply

10000 Plus + Power Supply

10000 Plus + Power Supply

10000 Plus + Power Supply

10000 Plus + Power Supply

10000 Plus + Power Supply

10000 Plus + Power Supply

10000 Plus + Power Supply

10000 Plus + Power Supply

10000 Plus + Power Supply

THE COPY DEADLINE FOR THE AUGUST ISSUE IS 31st

MAY. SERIES
DISCOUNTS ARE
AVAILABLE ON
SEMI-DISPLAY
BOOKINGS.

FOR SALE

essence

10000 Plus + Power Supply
10000 Plus + Power Supply
10000 Plus + Power Supply
10000 Plus + Power Supply
10000 Plus + Power Supply
10000 Plus + Power Supply
10000 Plus + Power Supply
10000 Plus + Power Supply
10000 Plus + Power Supply
10000 Plus + Power Supply

SPEEDY ASSEMBLER

Your Commodore's only own

Assembler, a 100% memory

resident program for loading

from tape or disk, featured in

the Your Commodore's

Machine Code Series and

now the N.C. standard

assembler only £8.95 tape or

disk. Order Code

YSASDSD (8pp) YSASDSD

(18p)

Order from Your Commodore

Readers Services at 9,

Mill Road, Hemel Hempstead

St Albans, Herts. AL9 9AT.

Order Code: YSASDSD. Please make

cheques payable to A.S.P.

Ltd., or telephone your

Access/Voice order on

0442 41221

CLASSIFIED COUPON

ALL CLASSIFIED ADVERTISEMENTS MUST BE PREPAID.
THERE ARE NO REIMBURSEMENTS FOR CANCELLATIONS.

I enclose my Cheque/Postal Order for £.....

insertions, made payable to Angus Specialist Publications

Group, at my request.

PLEASE SENT BY AIR/EXPRESS/POSTAL ORDER

EXP. DATE

FOR

POST CODE

DAYTIME TEL NO

Signature

Date

CLASSIFIED DEPT. ARDUGH HOUSE, BOUNDARY WAY,

HEMEL HEMPSTEAD HP2 7ST.

RATES: Lineage 58p per word (VAT inc) Semi-display: £10.95 +

VAT per single column cm minimum 3cm.

Series discounts available.

☐ FOR SALE ☐ SOFTWARE ☐ SPECIAL OFFERS ☐ OTHER STATE

Repairs Guide

0442
66850

CROYDON COMPUTER CENTRE

18 Burgess Road, Merton Road, Surrey, CR8 3JJ Tel: 01 883 3343

COMPUTER SERVICING

(In 4 years 1985)

We repair — on site or elsewhere — quick turnaround

- Commodore
- BBC & Ring-on (Approved Service Centres)
- Amstrad & Texas (Approved Service Centres)
- Disk Drives, Modems, Routers

Mail Orders by phone. Access & Visa accepted

COMMODORE REPAIRS

(with 1 year 100% guaranteed guarantee)

COMMODORE 88 £28.75

COMMODORE 128 £28.75

COMMODORE 160 £28.75

COMMODORE 200 £28.75

Price is all inclusive and we guarantee the whole computer for 1 year

0208 80000

24, Petrus Corner, Springfield, Chislehurst, Essex CM1 2EP
Tel: 0245 - 888343

COMMODORE SPARES & REPAIRS

(REPAIRS OFFERS)

884 814 £28.75
1284 £28.75
160 £28.75
200 £28.75

Price is all inclusive and we guarantee the whole computer for 1 year

COMMODORE 88 £28.75
COMMODORE 128 £28.75
COMMODORE 160 £28.75
COMMODORE 200 £28.75

Price is all inclusive and we guarantee the whole computer for 1 year

COMMODORE 88 £28.75
COMMODORE 128 £28.75
COMMODORE 160 £28.75
COMMODORE 200 £28.75

Price is all inclusive and we guarantee the whole computer for 1 year

COMMODORE 88 £28.75
COMMODORE 128 £28.75
COMMODORE 160 £28.75
COMMODORE 200 £28.75

Price is all inclusive and we guarantee the whole computer for 1 year

COMMODORE 88 £28.75
COMMODORE 128 £28.75
COMMODORE 160 £28.75
COMMODORE 200 £28.75

Price is all inclusive and we guarantee the whole computer for 1 year

COMMODORE 88 £28.75
COMMODORE 128 £28.75
COMMODORE 160 £28.75
COMMODORE 200 £28.75

Price is all inclusive and we guarantee the whole computer for 1 year

COMMODORE 88 £28.75
COMMODORE 128 £28.75
COMMODORE 160 £28.75
COMMODORE 200 £28.75

Price is all inclusive and we guarantee the whole computer for 1 year

COMMODORE 88 £28.75
COMMODORE 128 £28.75
COMMODORE 160 £28.75
COMMODORE 200 £28.75

Price is all inclusive and we guarantee the whole computer for 1 year

COMMODORE 88 £28.75
COMMODORE 128 £28.75
COMMODORE 160 £28.75
COMMODORE 200 £28.75

Price is all inclusive and we guarantee the whole computer for 1 year

COMMODORE 88 £28.75
COMMODORE 128 £28.75
COMMODORE 160 £28.75
COMMODORE 200 £28.75

Price is all inclusive and we guarantee the whole computer for 1 year

COMMODORE 88 £28.75
COMMODORE 128 £28.75
COMMODORE 160 £28.75
COMMODORE 200 £28.75

Price is all inclusive and we guarantee the whole computer for 1 year

PROBLEMS WITH YOUR COMMODORE ?
FOR FAST, RELIABLE AND PROFESSIONAL
REPAIRS AT COMPETITIVE PRICES

P M ENGINEERING

UNIT 282, MILTON SCIENCE
PARK, CAMBRIDGE CB4 4WE

ST. IVES (0223) 420007

We can also supply Hardware, Software, Disk Data,
Accessories and Spare parts.
CMA Repairs £25.00 inc. VAT (July 1985)

REPAIRS

AMIGA ATARI etc.
MONITORS & ALL ADD ON's
AMIGA UPGRADES AND MOD's

All estimates FREE

Prices from only £19.50

Call or post FAST TURNAROUND

PVS ELECTRONIC COMPONENTS

244 Deansgate Court,

Deansgate, Manchester M3 4BQ

Tel: 061 831 7086 Telex: 665595

Fax 061 832 6934



Computer Repairs

Instant Super Low Prices!

Instant of parts, labour and VAT



FAST AND
FOR
TECHNOLOGY

1 week turnaround! Collections/deliveries available for local area

• SUPER OFFERS •

SPECTRUM	512K inc. - Free Goods	ELECTRON	COMMODORE 154	£28
512K	512K inc. - Free Goods	COMMODORE 154	COMMODORE 154	£28
512K	512K inc. - Free Goods	COMMODORE 154	COMMODORE 154	£28
512K	512K inc. - Free Goods	COMMODORE 154	COMMODORE 154	£28
512K	512K inc. - Free Goods	COMMODORE 154	COMMODORE 154	£28

Please contact payment method. 1. Credit/debit card. 2. Cash. 3. Cheque. 4. Bank transfer. 5. Post office order. 6. Money order. 7. Personal cheque. 8. Company cheque. 9. Credit card. 10. Debit card. 11. Bank card. 12. Postcard. 13. Telegram. 14. Cable. 15. Telex. 16. Fax. 17. Email. 18. Internet. 19. Mobile phone. 20. Satellite phone. 21. Radio. 22. TV. 23. Video. 24. Audio. 25. Music. 26. Books. 27. Magazines. 28. Newspapers. 29. Journals. 30. Comics. 31. Games. 32. Toys. 33. Sports. 34. Hobbies. 35. Gardening. 36. Pets. 37. Travel. 38. Holidays. 39. Insurance. 40. Finance. 41. Law. 42. Medicine. 43. Science. 44. Technology. 45. Art. 46. Design. 47. Architecture. 48. Engineering. 49. Manufacturing. 50. Distribution. 51. Retail. 52. Wholesale. 53. Import. 54. Export. 55. Logistics. 56. Supply chain. 57. Procurement. 58. Sales. 59. Marketing. 60. Advertising. 61. Public relations. 62. Media. 63. Communications. 64. Information. 65. Research. 66. Development. 67. Innovation. 68. Creativity. 69. Problem solving. 70. Decision making. 71. Leadership. 72. Management. 73. Organization. 74. Planning. 75. Strategy. 76. Tactics. 77. Operations. 78. Logistics. 79. Distribution. 80. Sales. 81. Marketing. 82. Advertising. 83. Public relations. 84. Media. 85. Communications. 86. Information. 87. Research. 88. Development. 89. Innovation. 90. Creativity. 91. Problem solving. 92. Decision making. 93. Leadership. 94. Management. 95. Organization. 96. Planning. 97. Strategy. 98. Tactics. 99. Operations. 100. Logistics. 101. Distribution. 102. Sales. 103. Marketing. 104. Advertising. 105. Public relations. 106. Media. 107. Communications. 108. Information. 109. Research. 110. Development. 111. Innovation. 112. Creativity. 113. Problem solving. 114. Decision making. 115. Leadership. 116. Management. 117. Organization. 118. Planning. 119. Strategy. 120. Tactics. 121. Operations. 122. Logistics. 123. Distribution. 124. Sales. 125. Marketing. 126. Advertising. 127. Public relations. 128. Media. 129. Communications. 130. Information. 131. Research. 132. Development. 133. Innovation. 134. Creativity. 135. Problem solving. 136. Decision making. 137. Leadership. 138. Management. 139. Organization. 140. Planning. 141. Strategy. 142. Tactics. 143. Operations. 144. Logistics. 145. Distribution. 146. Sales. 147. Marketing. 148. Advertising. 149. Public relations. 150. Media. 151. Communications. 152. Information. 153. Research. 154. Development. 155. Innovation. 156. Creativity. 157. Problem solving. 158. Decision making. 159. Leadership. 160. Management. 161. Organization. 162. Planning. 163. Strategy. 164. Tactics. 165. Operations. 166. Logistics. 167. Distribution. 168. Sales. 169. Marketing. 170. Advertising. 171. Public relations. 172. Media. 173. Communications. 174. Information. 175. Research. 176. Development. 177. Innovation. 178. Creativity. 179. Problem solving. 180. Decision making. 181. Leadership. 182. Management. 183. Organization. 184. Planning. 185. Strategy. 186. Tactics. 187. Operations. 188. Logistics. 189. Distribution. 190. Sales. 191. Marketing. 192. Advertising. 193. Public relations. 194. Media. 195. Communications. 196. Information. 197. Research. 198. Development. 199. Innovation. 200. Creativity. 201. Problem solving. 202. Decision making. 203. Leadership. 204. Management. 205. Organization. 206. Planning. 207. Strategy. 208. Tactics. 209. Operations. 210. Logistics. 211. Distribution. 212. Sales. 213. Marketing. 214. Advertising. 215. Public relations. 216. Media. 217. Communications. 218. Information. 219. Research. 220. Development. 221. Innovation. 222. Creativity. 223. Problem solving. 224. Decision making. 225. Leadership. 226. Management. 227. Organization. 228. Planning. 229. Strategy. 230. Tactics. 231. Operations. 232. Logistics. 233. Distribution. 234. Sales. 235. Marketing. 236. Advertising. 237. Public relations. 238. Media. 239. Communications. 240. Information. 241. Research. 242. Development. 243. Innovation. 244. Creativity. 245. Problem solving. 246. Decision making. 247. Leadership. 248. Management. 249. Organization. 250. Planning. 251. Strategy. 252. Tactics. 253. Operations. 254. Logistics. 255. Distribution. 256. Sales. 257. Marketing. 258. Advertising. 259. Public relations. 260. Media. 261. Communications. 262. Information. 263. Research. 264. Development. 265. Innovation. 266. Creativity. 267. Problem solving. 268. Decision making. 269. Leadership. 270. Management. 271. Organization. 272. Planning. 273. Strategy. 274. Tactics. 275. Operations. 276. Logistics. 277. Distribution. 278. Sales. 279. Marketing. 280. Advertising. 281. Public relations. 282. Media. 283. Communications. 284. Information. 285. Research. 286. Development. 287. Innovation. 288. Creativity. 289. Problem solving. 290. Decision making. 291. Leadership. 292. Management. 293. Organization. 294. Planning. 295. Strategy. 296. Tactics. 297. Operations. 298. Logistics. 299. Distribution. 300. Sales. 301. Marketing. 302. Advertising. 303. Public relations. 304. Media. 305. Communications. 306. Information. 307. Research. 308. Development. 309. Innovation. 310. Creativity. 311. Problem solving. 312. Decision making. 313. Leadership. 314. Management. 315. Organization. 316. Planning. 317. Strategy. 318. Tactics. 319. Operations. 320. Logistics. 321. Distribution. 322. Sales. 323. Marketing. 324. Advertising. 325. Public relations. 326. Media. 327. Communications. 328. Information. 329. Research. 330. Development. 331. Innovation. 332. Creativity. 333. Problem solving. 334. Decision making. 335. Leadership. 336. Management. 337. Organization. 338. Planning. 339. Strategy. 340. Tactics. 341. Operations. 342. Logistics. 343. Distribution. 344. Sales. 345. Marketing. 346. Advertising. 347. Public relations. 348. Media. 349. Communications. 350. Information. 351. Research. 352. Development. 353. Innovation. 354. Creativity. 355. Problem solving. 356. Decision making. 357. Leadership. 358. Management. 359. Organization. 360. Planning. 361. Strategy. 362. Tactics. 363. Operations. 364. Logistics. 365. Distribution. 366. Sales. 367. Marketing. 368. Advertising. 369. Public relations. 370. Media. 371. Communications. 372. Information. 373. Research. 374. Development. 375. Innovation. 376. Creativity. 377. Problem solving. 378. Decision making. 379. Leadership. 380. Management. 381. Organization. 382. Planning. 383. Strategy. 384. Tactics. 385. Operations. 386. Logistics. 387. Distribution. 388. Sales. 389. Marketing. 390. Advertising. 391. Public relations. 392. Media. 393. Communications. 394. Information. 395. Research. 396. Development. 397. Innovation. 398. Creativity. 399. Problem solving. 400. Decision making. 401. Leadership. 402. Management. 403. Organization. 404. Planning. 405. Strategy. 406. Tactics. 407. Operations. 408. Logistics. 409. Distribution. 410. Sales. 411. Marketing. 412. Advertising. 413. Public relations. 414. Media. 415. Communications. 416. Information. 417. Research. 418. Development. 419. Innovation. 420. Creativity. 421. Problem solving. 422. Decision making. 423. Leadership. 424. Management. 425. Organization. 426. Planning. 427. Strategy. 428. Tactics. 429. Operations. 430. Logistics. 431. Distribution. 432. Sales. 433. Marketing. 434. Advertising. 435. Public relations. 436. Media. 437. Communications. 438. Information. 439. Research. 440. Development. 441. Innovation. 442. Creativity. 443. Problem solving. 444. Decision making. 445. Leadership. 446. Management. 447. Organization. 448. Planning. 449. Strategy. 450. Tactics. 451. Operations. 452. Logistics. 453. Distribution. 454. Sales. 455. Marketing. 456. Advertising. 457. Public relations. 458. Media. 459. Communications. 460. Information. 461. Research. 462. Development. 463. Innovation. 464. Creativity. 465. Problem solving. 466. Decision making. 467. Leadership. 468. Management. 469. Organization. 470. Planning. 471. Strategy. 472. Tactics. 473. Operations. 474. Logistics. 475. Distribution. 476. Sales. 477. Marketing. 478. Advertising. 479. Public relations. 480. Media. 481. Communications. 482. Information. 483. Research. 484. Development. 485. Innovation. 486. Creativity. 487. Problem solving. 488. Decision making. 489. Leadership. 490. Management. 491. Organization. 492. Planning. 493. Strategy. 494. Tactics. 495. Operations. 496. Logistics. 497. Distribution. 498. Sales. 499. Marketing. 500. Advertising. 501. Public relations. 502. Media. 503. Communications. 504. Information. 505. Research. 506. Development. 507. Innovation. 508. Creativity. 509. Problem solving. 510. Decision making. 511. Leadership. 512. Management. 513. Organization. 514. Planning. 515. Strategy. 516. Tactics. 517. Operations. 518. Logistics. 519. Distribution. 520. Sales. 521. Marketing. 522. Advertising. 523. Public relations. 524. Media. 525. Communications. 526. Information. 527. Research. 528. Development. 529. Innovation. 530. Creativity. 531. Problem solving. 532. Decision making. 533. Leadership. 534. Management. 535. Organization. 536. Planning. 537. Strategy. 538. Tactics. 539. Operations. 540. Logistics. 541. Distribution. 542. Sales. 543. Marketing. 544. Advertising. 545. Public relations. 546. Media. 547. Communications. 548. Information. 549. Research. 550. Development. 551. Innovation. 552. Creativity. 553. Problem solving. 554. Decision making. 555. Leadership. 556. Management. 557. Organization. 558. Planning. 559. Strategy. 560. Tactics. 561. Operations. 562. Logistics. 563. Distribution. 564. Sales. 565. Marketing. 566. Advertising. 567. Public relations. 568. Media. 569. Communications. 570. Information. 571. Research. 572. Development. 573. Innovation. 574. Creativity. 575. Problem solving. 576. Decision making. 577. Leadership. 578. Management. 579. Organization. 580. Planning. 581. Strategy. 582. Tactics. 583. Operations. 584. Logistics. 585. Distribution. 586. Sales. 587. Marketing. 588. Advertising. 589. Public relations. 590. Media. 591. Communications. 592. Information. 593. Research. 594. Development. 595. Innovation. 596. Creativity. 597. Problem solving. 598. Decision making. 599. Leadership. 600. Management. 601. Organization. 602. Planning. 603. Strategy. 604. Tactics. 605. Operations. 606. Logistics. 607. Distribution. 608. Sales. 609. Marketing. 610. Advertising. 611. Public relations. 612. Media. 613. Communications. 614. Information. 615. Research. 616. Development. 617. Innovation. 618. Creativity. 619. Problem solving. 620. Decision making. 621. Leadership. 622. Management. 623. Organization. 624. Planning. 625. Strategy. 626. Tactics. 627. Operations. 628. Logistics. 629. Distribution. 630. Sales. 631. Marketing. 632. Advertising. 633. Public relations. 634. Media. 635. Communications. 636. Information. 637. Research. 638. Development. 639. Innovation. 640. Creativity. 641. Problem solving. 642. Decision making. 643. Leadership. 644. Management. 645. Organization. 646. Planning. 647. Strategy. 648. Tactics. 649. Operations. 650. Logistics. 651. Distribution. 652. Sales. 653. Marketing. 654. Advertising. 655. Public relations. 656. Media. 657. Communications. 658. Information. 659. Research. 660. Development. 661. Innovation. 662. Creativity. 663. Problem solving. 664. Decision making. 665. Leadership. 666. Management. 667. Organization. 668. Planning. 669. Strategy. 670. Tactics. 671. Operations. 672. Logistics. 673. Distribution. 674. Sales. 675. Marketing. 676. Advertising. 677. Public relations. 678. Media. 679. Communications. 680. Information. 681. Research. 682. Development. 683. Innovation. 684. Creativity. 685. Problem solving. 686. Decision making. 687. Leadership. 688. Management. 689. Organization. 690. Planning. 691. Strategy. 692. Tactics. 693. Operations. 694. Logistics. 695. Distribution. 696. Sales. 697. Marketing. 698. Advertising. 699. Public relations. 700. Media. 701. Communications. 702. Information. 703. Research. 704. Development. 705. Innovation. 706. Creativity. 707. Problem solving. 708. Decision making. 709. Leadership. 710. Management. 711. Organization. 712. Planning. 713. Strategy. 714. Tactics. 715. Operations. 716. Logistics. 717. Distribution. 718. Sales. 719. Marketing. 720. Advertising. 721. Public relations. 722. Media. 723. Communications. 724. Information. 725. Research. 726. Development. 727. Innovation. 728. Creativity. 729. Problem solving. 730. Decision making. 731. Leadership. 732. Management. 733. Organization. 734. Planning. 735. Strategy. 736. Tactics. 737. Operations. 738. Logistics. 739. Distribution. 740. Sales. 741. Marketing. 742. Advertising. 743. Public relations. 744. Media. 745. Communications. 746. Information. 747. Research. 748. Development. 749. Innovation. 750. Creativity. 751. Problem solving. 752. Decision making. 753. Leadership. 754. Management. 755. Organization. 756. Planning. 757. Strategy. 758. Tactics. 759. Operations. 760. Logistics. 761. Distribution. 762. Sales. 763. Marketing. 764. Advertising. 765. Public relations. 766. Media. 767. Communications. 768. Information. 769. Research. 770. Development. 771. Innovation. 772. Creativity. 773. Problem solving. 774. Decision making. 775. Leadership. 776. Management. 777. Organization. 778. Planning. 779. Strategy. 780. Tactics. 781. Operations. 782. Logistics. 783. Distribution. 784. Sales. 785. Marketing. 786. Advertising. 787. Public relations. 788. Media. 789. Communications. 790. Information. 791. Research. 792. Development. 793. Innovation. 794. Creativity. 795. Problem solving. 796. Decision making. 797. Leadership. 798. Management. 799. Organization. 800. Planning. 801. Strategy. 802. Tactics. 803. Operations. 804. Logistics. 805. Distribution. 806. Sales. 807. Marketing. 808. Advertising. 809. Public relations. 810. Media. 811. Communications. 812. Information. 813. Research. 814. Development. 815. Innovation. 816. Creativity. 817. Problem solving. 818. Decision making. 819. Leadership. 820. Management. 821. Organization. 822. Planning. 823. Strategy. 824. Tactics. 825. Operations. 826. Logistics. 827. Distribution. 828. Sales. 829. Marketing. 830. Advertising. 831. Public relations. 832. Media. 833. Communications. 834. Information. 835. Research. 836. Development. 837. Innovation. 838. Creativity. 839. Problem solving. 840. Decision making. 841. Leadership. 842. Management. 843. Organization. 844. Planning. 845. Strategy. 846. Tactics. 847. Operations. 848. Logistics. 849. Distribution. 850. Sales. 851. Marketing. 852. Advertising. 853. Public relations. 854. Media. 855. Communications. 856. Information. 857. Research. 858. Development. 859. Innovation. 860. Creativity. 861. Problem solving. 862. Decision making. 863. Leadership. 864. Management. 865. Organization. 866. Planning. 867. Strategy. 868. Tactics. 869. Operations. 870. Logistics. 871. Distribution. 872. Sales. 873. Marketing. 874. Advertising. 875. Public relations. 876. Media. 877. Communications. 878. Information. 879. Research. 880. Development. 881. Innovation. 882. Creativity. 883. Problem solving. 884. Decision making. 885. Leadership. 886. Management. 887. Organization. 888. Planning. 889. Strategy. 890. Tactics. 891. Operations. 892. Logistics. 893. Distribution. 894. Sales. 895. Marketing. 896. Advertising. 897. Public relations. 898. Media. 899. Communications. 900. Information. 901. Research. 902. Development. 903. Innovation. 904. Creativity. 905. Problem solving. 906. Decision making. 907. Leadership. 908. Management. 909. Organization. 910. Planning. 911. Strategy. 912. Tactics. 913. Operations. 914. Logistics. 915. Distribution. 916. Sales. 917. Marketing. 918. Advertising. 919. Public relations. 920. Media. 921. Communications. 922. Information. 923. Research. 924. Development. 925. Innovation. 926. Creativity. 927. Problem solving. 928. Decision making. 929. Leadership. 930. Management. 931. Organization. 932. Planning. 933. Strategy. 934. Tactics. 935. Operations. 936. Logistics. 937. Distribution. 938. Sales. 939. Marketing. 940. Advertising. 941. Public relations. 942. Media. 943. Communications. 944. Information. 945. Research. 946. Development. 947. Innovation. 948. Creativity. 949. Problem solving. 950. Decision making. 951. Leadership. 952. Management. 953. Organization. 954. Planning. 955. Strategy. 956. Tactics. 957. Operations. 958. Logistics. 959. Distribution. 960. Sales. 961. Marketing. 962. Advertising. 963. Public relations. 964. Media. 965. Communications. 966. Information. 967. Research. 968. Development. 969. Innovation. 970. Creativity. 971. Problem solving. 972. Decision making. 973. Leadership. 974. Management. 975. Organization. 976. Planning. 977. Strategy. 978. Tactics. 979. Operations. 980. Logistics. 981. Distribution. 982. Sales. 983. Marketing. 984. Advertising. 985. Public relations. 986. Media. 987. Communications. 988. Information. 989. Research. 990. Development. 991. Innovation. 992. Creativity. 993. Problem solving. 994. Decision making. 995. Leadership. 996. Management. 997. Organization. 998. Planning. 999. Strategy. 1000. Tactics. 1001. Operations. 1002. Logistics. 1003. Distribution. 1004. Sales. 1005. Marketing. 1006. Advertising. 1007. Public relations. 1008. Media. 1009. Communications. 1010. Information. 1011. Research. 1012. Development. 1013. Innovation. 1014. Creativity. 1015. Problem solving. 1016. Decision making. 1017. Leadership. 1018. Management. 1019. Organization. 1020. Planning. 1021. Strategy. 1022. Tactics. 1023. Operations. 1024. Logistics. 1025. Distribution. 1026. Sales. 1027. Marketing. 1028. Advertising. 1029. Public relations. 1030. Media. 1031. Communications. 1032. Information. 1033. Research. 1034. Development. 1035. Innovation. 1036. Creativity. 1037. Problem solving. 1038. Decision making. 1039. Leadership. 1040. Management. 1041. Organization. 1042. Planning. 1043. Strategy. 1044. Tactics. 1045. Operations. 1046. Logistics. 1047. Distribution. 1048. Sales. 1049. Marketing. 1050. Advertising. 1051. Public relations. 1052. Media. 1053. Communications. 1054. Information. 1055. Research. 1056. Development. 1057. Innovation. 1058. Creativity. 1059. Problem solving. 1060. Decision making. 1061. Leadership. 1062. Management. 1063. Organization. 1064. Planning. 1065. Strategy. 1066. Tactics. 1067. Operations. 1068. Logistics. 1069. Distribution. 1070. Sales. 1071. Marketing. 1072. Advertising. 1073. Public relations. 1074. Media. 1075. Communications. 1076. Information. 1077. Research. 1078. Development. 1079. Innovation. 1080. Creativity. 1081. Problem solving. 1082. Decision making. 1083. Leadership. 1084. Management. 1085. Organization. 1086. Planning. 1087. Strategy. 1088. Tactics. 1089. Operations. 1090. Logistics. 1091. Distribution. 1092. Sales. 1093. Marketing. 1094. Advertising. 1095. Public relations. 1096. Media. 1097. Communications. 1098. Information. 1099. Research. 1100. Development. 1101. Innovation. 1102. Creativity. 1103. Problem solving. 1104. Decision making. 1105. Leadership. 1106. Management. 1107. Organization. 1108. Planning. 1109. Strategy. 1110. Tactics. 1111. Operations. 1112. Logistics. 1113. Distribution. 1114. Sales. 1115. Marketing. 1116. Advertising. 1117. Public relations. 1118. Media. 1119. Communications. 1120. Information. 1121. Research. 1122. Development. 1123. Innovation. 1124. Creativity. 1125. Problem solving. 1126. Decision making. 1127. Leadership. 1128. Management. 1129. Organization. 1130. Planning. 1131. Strategy. 1132. Tactics. 1133. Operations. 1134. Logistics. 1135. Distribution. 1136. Sales. 1137. Marketing. 1138. Advertising. 1139. Public relations. 1140. Media

B A E A H

Readers Problems

Though the Commodore 64 is one of the world's most popular microcomputers, it can be very difficult to find specific information about your particular machine.

At the Four Commodore office we receive literally hundreds of letters from you, our readers, on a wide range of subjects ranging from the simple 'Can you give me the telephone number for...', to the more complex 'I'm trying to write a program that uses a split screen. How do I do it?'

Unfortunately, the volume of mail received has become so great that it is impossible to answer every letter and still manage to publish a magazine each month.

For this reason we have felt it necessary to produce a number of guidelines for getting information from us:

- 1) We cannot guarantee to answer every letter sent to the magazine. Should it become apparent that a number of readers are suffering from the same problem, then we will reply to the letter via the Letter page.
- 2) A new helpline has been set up. This will be open for your queries on

Tuesday and Thursday afternoons between 2.00pm and 4.00pm. We will not be able to deal with our telephone queries at any other time. If our technical adviser is not available when you ring, then a message will be taken.

3) If you are having problems with one of our listings, can you please let us know in writing. This will enable us to see if a number of people are having the same problem. When a common problem becomes apparent with a program, then a correction sheet will be issued. Enclose a self-addressed, stamped envelope and we will send you a copy of the correction sheet as soon as it is available.

We are sorry that it has become necessary to instigate these rules. However, we are sure that you will agree with us that the more time that we can spend making Four Commodore the most informative magazine around, the better.

For program queries write to:
Program Corrections
Four Commodore
Apex House,
Boundary Way,
Harold Hempstead
HPI 781
Tel: 0442-66251

CORRECTIONS FOR MARCH '89

READ FOR HOME

Once again those intractable grammars have caused havoc.

The program as it stands will not run correctly. This is due to the fact that quite a bit of code is missing from the loader program. An amendment is on its way. The update will be published as soon as it reaches our office. We apologise for any inconvenience to our Plus 4 readers.

CORRECTIONS FOR APRIL '89

BALANCE SHEET

Unfortunately there appeared a couple of errors in the Balance Sheet program from the April Edition.

- 1) Page 31, right hand column, paragraph 4 should read Load "LIST4" and not Load "LIST1".
- 2) The second line of Listing 1 should be POKET70,131:POKET71,164
- 3) All the REM statements should be taken out of Listing 1.
- 4) Again on page 31, right hand column, paragraph 1. This should read:
50:SY557812:G=SHEET:AJ

Commodore Where Are You?

At the Four Commodore office we are repeatedly asked for the address and telephone number of Commodore

U.K. Many people, after referring to their computer manuals, believe them to be based in Carby.

The Commodore plant at Carby was closed down some time ago. Reproduced here you will find the correct

address for Commodore U.K.
Commodore Business Machines, (UK),
Commodore House, The Switchback,
Barks SLE 7XA,
Tel: (0625) 770888
Gardiner Road, Molebushard.

The Nibbles By Alan Batchelor



AMIGA

ATARI
ST

IBM PC

APPLE
MAC

THE SPECIALIST
MUSIC MAGAZINE
FOR COMPUTER USERS

MUSIC X-The Review

MICRO MUSIC

Find out about the magazine
the computer music world has been
waiting for

KCS-Just what the
Doctor orders!!

ACORN

DANBORN
64

AMSTRAD
CPC

ZX
SPECTRUM

THE TIME HAS COME . . .

Britain's First Music Magazine for the Computer User!

Featuring Reviews on Hardware & Software across all formats - Micro Music is the magazine the Computer Musician has been waiting for. Available 10th March from all good Newsagents

Emlyn Hughes



INTERNATIONAL



Quite simply the best football simulation available for the 64. Nothing short of superb.

ZZAP 64

**AVAILABLE NOW FOR
COMMODORE 64
£9.95 tape £12.95 disk**
Coming soon for Spectrum & Amstrad

**GAME
OF THE YEAR!**

**WINNER OF CCI OSCARS FOR
BEST SPORTS SIMULATION
AND BEST GAME OF THE YEAR!**

Audiogenic

Audiogenic Software Limited, Winchester House,
Canning Road, Harrow HA3 7SL, England

Order by phone on 01 861 1166

